MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8381– DATASTRUCTURES LABORATORY

R 2017
LAB MANUAL

INDEX

S.No.	Name of the Experiment	Page No	Date of Experiment	Remarks
1a	Stack-Array Implementation			
1b	Queue-Array Implementation			
2	List-Array Implementation			
3a	List-Linked List Implementation			
3b	Stack-Linked List Implementation			
3c	Queue-Linked List Implementation			
4a	Polynomial Addition			
4b	Infix to Postfix Conversion			
4c	Evaluation Postfix Expression			
5	Binary Tree			
6	Binary Search Tree			
7	AVL Tree			
8	Heap using Priority Queue			
9	Graph Traversal			
10	Topological Sort			
11a	Linear & Binary Search			
11b	Sorting			
12	Linear Probing			

Ex.No.-1a STACK – ARRAY IMPLEMENTATION

AIM:

To write a C program to implement the stack using arrays.

ALGORITHM:

(i) Push Operation:

- To push an element into the stack, check whether the top of the stack is greater than or equal to the maximum size of the stack.
- If so, then return stack is full and element cannot be pushed into the stack.
- Else, increment the top by one and push the new element in the new position of top.

(ii) Pop Operation:

- To pop an element from the stack, check whether the top of the stack is equal to -1.
- If so, then return stack is empty and element cannot be popped from the stack. ○ Else, decrement the top by one.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-1b QUEUE – ARRAY IMPLEMENTATION

AIM:

To write a C program to implement the queue using arrays.

ALGORITHM:

Enqueue:

1. Check if queue is not full.
2. If it is full then return queue overflow and item cannot be inserted.
3. If not, check if rear value is -1, if so then increment rear and by 1; if not increment front by 1.
4. Store the item in the new value of front.

Dequeue:

1. Check if queue is not empty.
2. If it is empty, return queue underflow and dequeue operation cannot be done.
3. If queue is not empty , check if rear and front are equal.
    - If so assign -1 to front and rear.
    - If not decrement front by 1.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-2 LIST – ARRAY IMPLEMENTATION

AIM:

To write a C program to implement the List using arrays.

ALGORITHM:

1. Start the program.
2. Read the number of elements in the list and create the list.
3. Read the position and element to be inserted.
4. Adjust the position and insert the element.
5. Read the position and element to be deleted.
6. Remove the element from the list and adjust the position.
7. Read the element to be searched.
8. Compare the elements in the list with searching element.
9. If element is found, display it else display element is not found.
10. Display all the elements in the list.
11. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-3a LIST – LINKED LIST IMPLEMENTATION

AIM:

To write a C program to implement the List ADT using linked list.

ALGORITHM:

1. Start the program.
2. Create a node using structure
3. Dynamically allocate memory to node
4. Create and add nodes to linked list.
5. Read the element to be inserted.
6. Insert the elements into the list.
7. Read the element to be deleted.
8. Remove that element and node from the list.
9. Adjust the pointers.
10. Display the OUTPUT:

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-3b STACK – LINKED LIST IMPLEMENTATION

AIM:

To write a C program to implement the stack using linked list.

ALGORITHM:

A) Push Operation:

1. To push an element into the stack, copy the element to be inserted in the data field of the new node.
2. Assign the reference field of the new node as NULL.
3. Check if top is not NULL, if so, then assign the value of top in the reference field of new node.
4. Assign the address of the new node to the top.

B) Pop Operation:

1. To pop an element from the stack, check whether the top of the stack is NULL.
2. If so, then return stack is empty and element cannot be popped from the stack.
3. Else, assign the top value to a temporary node.
4. Now assign the value in the reference field of the node pointed by top to the top value.
5. Return the value in the data field of the temporary node as the element deleted and delete the temporary node.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-3c QUEUE – LINKED LIST IMPLEMENTATION

AIM:

To write a C program to implement the queue using linked list.

ALGORITHM:

Enqueue:

1. Create a new node and allocate memory space for the new node.
2. Assign the element to be inserted in the data field of the new node.
3. Assign NULL to the address field of the new node.
4. Check if rear and front pointers are NULL.
5. If so, then make the front and rear pointers to point to new node.
6. If not, then assign address of the new node as the rear pointer

value. Dequeue:

1. Check if queue is not empty.
2. If it is empty, return queue underflow and dequeue operation cannot be done.
3. If not, assign the front->next value as the new front pointer and free the deleted node.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-4a POLYNOMIAL ADDITION

AIM:

To write a C program to perform polynomial addition using linked list.

ALGORITHM:

1. Start the program.
2. Declare the node as link.
3. Allocate memory space for pol1, pol2 and poly.
4. Read pol1 and pol2.
5. Call the function polyadd.
6. Add the co-efficients of poly1 and poly2 having the equal power and store it in poly.
7. If there is no co-efficient having equal power in poly1 and poly2, then add it to poly.
8. Display the poly1, poly2 and poly.
9. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-4b INFIX TO POSTFIX CONVERSION

AIM:

To write a C program to perform infix to postfix conversion using stack.

ALGORITHM:

1. Define a stack
2. Go through each character in the string
3. If it is between 0 to 9, append it to output string.
4. If it is left brace push to stack
5. If it is operator *+-/ then
    a. If the stack is empty push it to the stack
    b. If the stack is not empty then start a loop:
        i. If the top of the stack has higher precedence
        ii. Then pop and append to output string
        iii. Else break
        iv. Push to the stack
6. If it is right brace then
    a. While stack not empty and top not equal to left brace
    b. Pop from stack and append to output string
    c. Finally pop out the left brace.
7. If there is any input in the stack pop and append to the output string.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-4c EVALUATING POSTFIX EXPRESSION

AIM:

To write a C program to evaluate postfix expression using stack.

ALGORITHM:

1. Start the program.
2. Scan the Postfix string from left to right.
3. Initialise an empty stack.
4. If the scannned character is an operand, add it to the stack. If the scanned character is an operator, there will be atleast two operands in the stack.
5. If the scanned character is an Operator, then we store the top most element of the stack(topStack) in a variable temp. Pop the stack. Now evaluate topStack(Operator)temp. Pop the stack and Push result into the stack.
6. Repeat this step till all the characters are scanned.
7. After all characters are scanned, we will have only one element in the stack. Return topStack.
8. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-5 BINARY TREE

AIM:

To write a C program to implement binary tree and its traversals.

ALGORITHM:

1. Start the program.
2. Declare the node.
3. Create the binary tree by inserting elements into it.
4. Traverse the binary tree by inorder and display the nodes.
5. Traverse the binary tree by preorder and display the nodes.
6. Traverse the binary tree by postorder and display the nodes.
7. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-6 BINARY SEARCH TREE

AIM:

To write a C program to implement binary search tree.

ALGORITHM:

1. Start the program.
2. Declare the node.
3. Read the elements to be inserted.
4. Create the binary search tree.
5. Read the element to be searched.
6. Visit the nodes by inorder.
7. Find the searching node and display if it is present with parent node.
8. Read the element to be removed from BST.
9. Delete that node from BST.
10. Display the binary search tree by inorder.
11. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-7 AVL TREE

AIM:

To write a C program to implement an AVL tree.

ALGORITHM:

1. Start the program.
2. Declare the node.
3. Read the elements and create a tree.

Insert:

4. Insert a new node as new leaf node just as in ordinary binary search tree.
5. Now trace the path from inserted node towards root. For each node, „n‟ encountered , check if heights of left(n) and right(n) differ by atmost 1.
    a) if yes, move towards parent(n).
    b) Otherwise restructure the by doing either a single rotation or a double rotation.

Delete:

6. Search the node to be deleted.
7. If the node to be deleted is a leaf node, then simply make it NULL to remove.
8. If the node to be deleted is not a leaf node, then the node must be swapped with its inorder successor. Once the node is swapped, then remove the node.
9. Traverse back up the path towards root, check the balance factor of every node along the path.
10. If there is unbalanced in some subtree then balance the subtree using appropriate single or double rotation.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-8 HEAP USING PRIORITY QUEUE

AIM:

To write a C program to implement heap sort using priority queue.

ALGORITHM:

1. Start the program.
2. Read the elements to be inserted in heap.
3. Insert the element one by one.
4. Construct the heap structure to satisfy heap property of maxheap.
5. Display the heapified structure.
6. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-9 GRAPH TRAVERSAL

AIM:

To write a C program to implement graph traversals by Breadth First Search and Depth First Search.

ALGORITHM:

Breadth First Search:

1. Start the program.
2. Read the number of vertices and adjacency matrix.
3. Read the vertex from which to traverse the graph.
4. Initialize the visited array to 1 and insert the visited vertex in the queue.
5. Visit the vertex which is at the front of the queue.
6. Delete it from the queue and place its adjacent nodes in the queue.
7. Repeat the steps 5 & 6 , till the queue is not empty.
8. Display the traversal path.
9. Stop the program.

Depth First Search:

1. Start the program.
2. Read the number of vertices and adjacency matrix.
3. Initialize the visited array to 1.
4. Traverse the path one by one and push the visited vertex in the stack.
5. When there is no vertex further, we traverse back and search for unvisited vertex.
6. Display the traversal path.
7. Stop the program.

a) Breadth First Search

PROGRAM:

b) Depth First Search

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-10 TOPOLOGICAL SORTING

AIM:

To write a C program to perform topological sorting (Application of a graph).

ALGORITHM:

1. Start the program.
2. Read the number of vertices and adjacency matrix of a graph.
3. Find a vertex with no incoming edges.
4. Delete it along with all the edges outgoing from it.
5. If there are more than one such vertices then break the tie randomly.
6. Store the vertices that are deleted.
7. Display these vertices that give topologically sorted list.
8. Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-11a LINEAR & BINARY SEARCH

AIM:

To write a C program to perform linear search and binary search.

ALGORITHM:

Linear Search

1. Read n numbers and search value.
2. If search value is equal to first element then print value is found.
3. Else search with the second element and so on.

Binary Search

1. Read n numbers and search value.
2. If search value is equal to middle element then print value is found.
3. If search value is less than middle element then search left half of list with the same method.
4. Else search right half of list with the same method.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-11b SORTING

AIM:

To write a C program to perform insertion sort, quick sort and bubble sort**.**

ALGORITHM:

Insertion Sort

1. Get the n elements to be sorted.
2. The ith element is compared from (i-1)th to 0th element and placed in properposition according to ascending value.
3. Repeat the above step

until the last element.Quick Sort

1. Pick an element, called a pivot, from the list.
2. Reorder the list so that all elements which are less than the pivot come before thepivot and so that all elements greater than the pivot come after it.
3. After this partitioning, the pivot is in its final position. This is called the partitionoperation.
4. Recursively sort the sub-list of lesser elements and the sub-list of greater elementsBubble Sort

1. Get the n elements to be sorted.
2. Compare the first two elements of the array and swap if necessary.
3. Then, again second and third elements are compared and swapped if it is necessary andcontinue this process until last and second last element is compared and swapped.
4. Repeat the above two steps n-1 times and print the result.

PROGRAM:

OUTPUT:

RESULT:

Ex.No.-12 LINEAR PROBING

AIM:

To write a c program to create hash table and collision handling by linear probing.

ALGORITHM:

1. Start the program.
2. Read the numbers to be stored in hash table.
3. Create the hash function by generating the hash key.
4. If the location indicated by hash key is empty, then place the number in the hash table.
5. If collision occurs, then search for empty location.
6. If found, place the number at that location.
7. Display the hash table.

PROGRAM:

OUTPUT:

RESULT:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8383– OBJECT ORIENTED PROGRAMMING LABORATORY

R 2017
LAB MANUAL

INDEX

EX. NO : 1 GENERATING ELECTRICITY BILL

AIM:

To Develop a Java application to generate Electricity bill.

ALGORITHM:

1. Import the java packages.
2. Create a class with members Consumer no., consumer name, previous month reading, current month reading, type of EB connection (i.e domestic or commercial).
3. Class also contains methods domesticbillcalc and commercialbillcalc with its parameters to compute bill amount.
4. Check whether the type of connection is domestic or commercial.
   If domestic, calculate the bill amount as follows:
   First 100 units - Rs. 1 per unit
   101-200 units - Rs. 2.50 per unit
   201 -500 units - Rs. 4 per unit
   > 501 units - Rs. 6 per unit
   If commercial, calculate the bill amount as follows:
   First 100 units - Rs. 2 per unit
   101-200 units - Rs. 4.50 per unit
   201 -500 units - Rs. 6 per unit
   > 501 units - Rs. 7 per unit
5. Calculate the units consumed by finding the differences between previous month reading and current month reading.
6. By using Scanner class get the input during runtime.
7. Create object for a class in memory and assign it to the reference variable, then the method is invoked.
8. Finally, the bill amount is displayed based on type of connection.

PROGRAM:

OUTPUT:

RESULT:

EX. NO : 2 CURRENCY CONVERTER, DISTANCE CONVERTER AND TIME CONVERTER USING PACKAGES

AIM:

To develop a java application to implement currency converter, distance converter and time converter using packages.

ALGORITHM:

1. The package keyword is used to create a package in java.
2. Create a class CurrencyConverter inside a package name CurrencyConverter.
3. Class also contains methods dollortoinr, inrtodollor, eurotoinr, inrtoeuro, yentoinr, and inrtoyen with its parameters to convert given currency.
4. Create a class DistanceConverter inside a package name DistanceConverter.
5. Class also contains methods metertokm, kmtometer, milestokm and kmtomiles with its parameters to convert given distance.
6. Create a class TimeConverter inside a package name TimeConverter.
7. Class also contains methods hourstominutes, minutestohours, hourstoseconds and secondstohours with its parameters to convert given time.
8. Import the CurrencyConverter, DistanceConverter, TimeConverter and other java packages.
9. Create a class Converter and object for a class in memory and assign it to the reference variable, then the method is invoked.
10. By using Scanner class get the choices for switch statement during runtime.
11. By using switch case statement we can convert currency, distance and time for each choice.
12. Create object for a class in memory and assign it to the reference variable, then the method is invoked.
13. Finally, the conversion is displayed based on type of converter.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 3 GENERATING EMPLOYEE PAYROLL DETAILS

AIM:

To develop a java application for generating pay slips of employees with their gross and net salary.

ALGORITHM:

1. The package keyword is used to create a package in java.
2. Create a class Employee inside a package name employee.
3. Class Employee contains Emp_name, Emp_id, Address, Mail_id, Mobile_no as members.
4. By using Constructor initialize the instance variable of Employee class and display method is used to print employee details.
5. Create classes Programmer, AssistantProfessor, AssociateProfessor and Professor that extends Employee class and define necessary constructor for sub classes.
6. Each sub classes has its own instance variable like bPay and des.
7. Override the paySlip method in each sub classes to calculate the gross and net salary
8. By using super () method subclasses initialize the super class constructor.
9. Import employee package and create the object for Empolyee class.
10. Create different Employee object to add ArrayList<> classes.
11.DisplayEmployee method is used to display all employee playSlip details

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 4 DESIGN A JAVA INTERFACE FOR ADT STACK

AIM:

To Design a Java interface for ADT Stack and implement this interface using array, provide necessary exception handling in the implementation.

ALGORITHM:

1. Import the java packages.
2. Design an interface for MyStack with functions push, pop and display.
3. Define a class StackArray to implement the MyStack using array.
4. Define the functions of the interface accordingly and handle the stack overflow and underflow exceptions.
5. Create a class StackAdt and object for a class StackArray in memory and assign it to the reference variable, then the method is invoked.
6. By using Scanner class get the choices for switch statement during runtime.
7. By using switch case statement we can push, pop and display the elements for each choice.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 5 STRING OPERATIONS USING ARRAYLIST

AIM:

To write a java program to perform string operations using ArrayList.

ALGORITHM:

1. Import the java packages.
2. Define a class ArrayList and perform following functions:
   a. Append - add at end
   b. Insert - add at particular index
   c. Search
   d. List all string starts with given letter
3. Create an object for ArrayList to add string elements.
4. By using ArrayList Method – add the elements are added into the Array List, the new element gets added after the last element unless the index is specified.
5. The elements in ArrayList are displayed.
6. Insert the specified element at the specified position index in this list.
7. After inserting the elements in ArrayList, the elements are displayed.
8. Search an object in ArrayList whether it is listed under this instance or not.
9. Finally List all string starts with given letter in ArrayList and displays the elements.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 6 FINDING THE AREA OF DIFFERENT SHAPES

AIM:

To write a java program to find the area of different shapes by using abstract class.

ALGORITHM:

1. Import the java packages.
2. Create an abstract class named Shape that contains two integers and an empty method named printArea().
3. Create a class Rectangle that extends the class Shape. Override the method printArea () by getting Width and Length then compute the area and prints the area of the Rectangle.
4. Create a class Triangle that extends the class Shape. Override the method printArea () by getting Base and Height then compute the area and prints the area of the Triangle.
5. Create a class Circle that extends the class Shape. Override the method printArea () by getting the Radius, then compute the area and prints the area of the Circle.
6. By using Scanner class get the input during runtime.
7. Create object for a class in memory and assign it to the reference variable, then the method is invoked.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 7 CREATING OWN EXCEPTIONS

AIM:

To write a java program to implement user defined exception handling.

ALGORITHM:

1. Import the java packages.
2. Create a subclass of Exception named as MyException it has only a constructor plus an overloaded toString ( ) method that displays the value of the exception.
3. The exception is thrown when compute ( ) integer parameter is greater than 10.
4. The main ( ) method sets up an exception handler for MyException, then calls compute ( ) with a legal value (less than 10) and an illegal one to show both paths through the code.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 8 GETTING FILE INFORMATION

AIM:

To write a java program to implement file information such as reads a file name from the user, displays information about whether the file exists, whether the file is readable, or writable, the type of file and the length of the file in bytes.

ALGORITHM:

1. Import the java packages.
2. By using Scanner class get the input during runtime.
3. By using File class method create a File object associated with the file or directory specified by pathname. The pathname can contain path information as well as a file or directory name.
4. The exists() checks whether the file denoted by the pathname exists. Returns true if and only if the file denoted by the pathname exists; false otherwise
5. The getAbsolutePath() returns the absolute pathname string of the pathname.
6. The canRead() checks whether the application can read the file denoted by the pathname. Returns true if and only if the file specified by the pathname exists and can be read by the application; false otherwise.
7. The canWrite() checks whether the application can modify to the file denoted by the pathname. Returns true if and only if the file system actually contains a file denoted by the pathname and the application is allowed to write to the file; false otherwise.
8. The length() returns the length of the file denoted by the pathname. The return value is unspecified if the pathname denotes a directory.
9. The endsWith() returns true if the given string ends with the string given as argument for the method else it returns false.
10. The program uses conditional operator to check different functionalities of the given file.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 9 MULTI THREADED APPLICATION

AIM:

To write a program that implements a multi-threaded application that has three threads. First thread generates a random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.

ALGORITHM:

1. Import the java packages.
2. Create a thread that generates random number, Obtain one random number and check is odd or even.
3. If number is even then create and start thread that computes square of a number, Compute number * number and display the answer.
4. Notify to Random number thread and goto step 7.
5. If number is odd then create and start thread that computes cube of a number, Compute number * number * number and display the answer.
6. Notify to Random number thread and goto step 7.
7. Wait for 1 Second and Continue to Step 3 until user wants to exits.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 10 GENERIC PROGRAMMING

AIM:

To write a java program to find the maximum value from the given type of elements using a generic function.

ALGORITHM:

1. Import the java packages.
2. Comparable interface is used to order the objects of user-defined class.
3. This interface is found in java.lang package and contains only one method named compareTo(Object).
4. The compareTo() method works by returning an int value that is either positive, negative, or zero.
5. Create a generic method max(), that can accept any type of argument.
6. Then sets the first element as the max element, and then compares all other elements with the max element using compareTo() method
7. Finally the function returns an element which has the maximum value.
8. We can call generic method by passing with different types of arguments, the compiler handles each method.

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 11 EVENT - DRIVEN PROGRAMMING

AIM:

To write a java program to design a calculator using event-driven programming paradigm of Java with the following options.

a) Decimal manipulations

b) Scientific manipulations

ALGORITHM:

1. Import the java packages.
2. Create the class calculator by implementing the class Jframe and interface actionListener.
3. Declare the buttons required using JButton.
4. Design the layout of the calculator using the setLayout,textpanel(), Panel(),Jtextfield(),setfont() methods.
5. Define the actions to be performed for each key using ActionListener.
6. Enable the scientific or standard calculator using the method method add().
7. Define the mathematical operations to be performed for the mathematical symbols.
8. Select the required mathematical operations using switch as the calculator.
9. Pass the parameters for the methods used.
10. Make the frame visible by using the method setVisible().

PROGRAM:

OUTPUT:

RESULT:

EX. NO. 12 MINI PROJECT - OPAC SYSTEM

AIM:

To develop a mini project OPAC system for library using Java concepts.

ALGORITHM:

1. Import the awt,swing packages.
2. Extend the JFrame which implements actionlistener to the class datas.
3. Create the textfield for id, name and button for next, address and the panel.
4. Create object for the getcontentpane().
5. Assign the length and breadth value for the layout using gridlayout.
6. Add the new labels for ISBN and book name.
7. Add the new button for the nextbook
8. Create the bookname under the driver jdbc odbc driver in the try block.
9. Create the object for exception as e and use it for catching the error.
10. Show all the records using showrecord.

PROGRAM:

OUTPUT:

RESULT:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8461– OPERATING SYSTEMS LABORATORY

R 2017
LAB MANUAL

INDEX

S.No.	Name of the Experiment	Page No	Date of Experiment	Remarks
1.	Basics of UNIX commands			
2.	Write programs using the following system calls of UNIX operating system fork, exec, getpid, exit, wait, close, stat, opendir, readdir			
3.	Write C programs to simulate UNIX commands like cp, ls, grep, etc.			
4.	Shell Programming			
5.	Write C programs to implement the various CPU Scheduling Algorithms			
6.	Implementation of Semaphores			
7.	Implementation of Shared memory and IPC			
8.	Bankers Algorithm for Deadlock Avoidance			
9.	Implementation of Deadlock Detection Algorithm			
10.	Write C program to implement Threading & Synchronization Applications			
11	Implementation of the following Memory Allocation Methods for fixed partition a) First Fit b) Worst Fit c) Best Fit			
12	Implementation of Paging Technique of Memory Management			
13	Implementation of the following Page Replacement Algorithms a) FIFO b) LRU c) LFU			

14	Implementation of the various File Organization Techniques			
15	Implementation of the following File Allocation Strategies a) Sequential b) Indexed c) Linked			

Ex.No:1.a	BASICS OF UNIX COMMANDS
	INTRODUCTION TO UNIX

AIM:

To study about the basics of UNIX

UNIX:

It is a multi-user operating system. Developed at AT & T Bell Industries, USA in 1969.

Ken Thomson along with Dennis Ritchie developed it from MULTICS (Multiplexed Information and Computing Service) OS.

By1980, UNIX had been completely rewritten using C language.

LINUX:

It is similar to UNIX, which is created by Linus Torualds. All UNIX commands works in Linux. Linux is a open source software. The main feature of Linux is coexisting with other OS such as windows and UNIX.

STRUCTURE OF A LINUXSYSTEM:

It consists of three parts.

a) UNIX kernel
b) Shells
c) Tools and Applications

UNIX KERNEL:

Kernel is the core of the UNIX OS. It controls all tasks, schedule all Processes and carries out all the functions of OS.

Decides when one programs tops and another starts.

SHELL:

Shell is the command interpreter in the UNIX OS. It accepts command from the user and analyses and interprets them

Ex.No:1.b	BASICS OF UNIX COMMANDS
	BASIC UNIX COMMANDS

AIM:

To study of Basic UNIX Commands and various UNIX editors such as vi, ed, ex and EMACS.

CONTENT:

Note: Syn->Syntax

a) date

–used to check the date and time

Syn:$date

Format	Purpose	Example	Result
+%m	To display only month	$date+%m	06
+%h	To display month name	$date+%h	June
+%d	To display day of month	$date+%d	O1
+%y	To display last two digits of years	$date+%y	09
+%H	To display hours	$date+%H	10
+%M	To display minutes	$date+%M	45
+%S	To display seconds	$date+%S	55

b) cal

–used to display the calendar

Syn:$cal 2 2009

c)echo

–used to print the message on the screen.

Syn:$echo "text"

d) ls

–used to list the files. Your files are kept in a directory.

Syn:$lsls–s

All files (include files with prefix)

ls–l Lodetai (provide file statistics)

ls–t Order by creation time

ls– u Sort by access time (or show when last accessed together with –l)

ls–s Order by size

ls–r Reverse order

ls–f Mark directories with /,executable with* , symbolic links with @, local sockets with =, named pipes(FIFOs)with

ls–s Show file size

ls– h" Human Readable", show file size in Kilo Bytes & Mega Bytes (h can be used together with –l or)

ls[a-m]*List all the files whose name begin with alphabets From „a" to „m"ls[a]*List all the files whose name begins with „a" or „A"
Eg:$ls>my list Output of „ls" command is stored to disk file named „my list"

e)lp
–used to take printouts
Syn:$lp filename

f)man
–used to provide manual help on every UNIX commands.
Syn:$man unix command
$man cat

g)who & **whoami**
–it displays data about all users who have logged into the system currently. The next command displays about current user only.
Syn:whowhoami

h) uptime
–tells you how long the computer has been running since its last reboot or power-off.
Syn:$uptime

i)uname
–it displays the system information such as hardware platform, system name and processor, OS type.
Syn:$uname–a

j) hostname
–displays and set system host name
Syn:$ hostname

k) bc
–stands for „best calculator"

```
$bc
10/2*3
15
```

```
$ bc
scale =1
2.25+1
3.35
quit
```

```
$ bc
ibase=2
obase=16
11010011
89275
1010
Ā
Quit
```

```
$ bc
sqrt(196)
14 quit
```

```
$bc
for(i=1;i<3;i=i+1)I
1
2
3 quit
```

```
$ bc-l
scale=2
s(3.14)
0
```

FILE MANIPULATION COMMANDS

a) **cat**–this create, view and concatenate files.

Creation:

Syn:$cat>filename

Viewing:

Syn:$cat filename

Add text to an existing file:

Syn:$cat>>filename

Concatenate:

Syn:$catfile1file2>file3

$catfile1file2>>file3 (no over writing of file3)

b) **grep**–used to search a particular word or pattern related to that word from the file.

Syn:$grep search word filename

Eg:$grep anu student

c) **rm**–deletes a file from the file system

Syn:$rm filename

d) **touch**–used to create a blank file.

Syn:$touch file names

e) **cp**–copies the files or directories

Syn:$cpsource file destination file

Eg:$cp student stud

f) **mv**–to rename the file or directory

syn:$mv old file new file

Eg:$mv–i student student list(-i prompt when overwrite)

g) **cut**–it cuts or pickup a given number of character or fields of the file.

Syn:$cut<option><filename>

Eg: $cut –c filename

$cut–c1-10emp

$cut–f 3,6emp

$ cut –f 3-6 emp

-c cutting columns

-f cutting fields

h) **head**–displays10 lines from the head(top)of a given file

Syn:$head filename

Eg:$head student

To display the top two lines:

Syn:$head-2student

i) **tail**–displays last 10 lines of the file
Syn:$tail filename
Eg:$tail student
To display the bottom two lines;
Syn:$ tail -2 student

j) **chmod**–used to change the permissions of a file or directory.
Syn:$ch mod category operation permission file
Where, Category–is the user type
Operation–is used to assign or remove permission
Permission–is the type of permission
File–are used to assign or remove permission all

Examples:
$chmodu-wx student
Removes write and execute permission for users
$ch modu+rw,g+rwstudent
Assigns read and write permission for users and groups
$chmodg=rwx student
Assigns absolute permission for groups of all read, write and execute permissions

k) **wc**–it counts the number of lines, words, character in a specified file(s)
with the options as –l,-w,-c

Category	Operation	Permission
u– users g–group o– others	+assign -remove =assign absolutely	r– read w– write x-execute

Syn: $wc –l filename
$wc –w filename
$wc–c filename

Ex.No:1.c	BASICS OF UNIX COMMANDS
	UNIX EDITORS

AIM:

To study of various UNIX editors such as vi, ed, ex and EMACS.

CONCEPT:

Editor is a program that allows user to see a portions a file on the screen and modify characters and lines by simply typing at the current position. UNIX supports variety of Editors. They are:

ed ex vi
EMACS

Vi- vi is stands for "visual".vi is the most important and powerful editor.vi is a full screen editor that allows user to view and edit entire document at the same time.vi editor was written in the University of California, at Berkley by Bill Joy, who is one of the co-founder of Sun Microsystems.

Features of vi:

It is easy to learn and has more powerful features.
Itworksgreatspeedandiscasesensitive.vihaspowerfulundofunctionsandhas3modes:

1. Command mode
2. Insert mode
3. Escape or ex mode

In command mode, no text is displayed on the screen.
In Insert mode, it permits user to edit insert or replace text.
In escape mode, it displays commands at command line.
Moving the cursor with the help of h, l, k, j, I, etc

EMACS Editor

Motion Commands:
M-> Move to end of file
M-< Move to beginning of file
C-v Move forward a screen M –v Move
backward a screen C –n Move to next line
C-p Move to previous line
C-a Move to the beginning of the line
C-e Move to the end of the line
C-f Move forward a character
C-b Move backward a character
M-f Move forward a word
M-b Move backward a word

Deletion Commands:

DEL	delete the previous character C -d
	delete the current character M -DEL
	delete the previous word
M-d	delete the next word
C-x DEL	deletes the previous sentence
M-k	delete the rest of the current sentence
C-k	deletes the rest of the current line
C-xu	undo the lasted it change

Search and Replace in EMACS:

y	Change the occurrence of the pattern
n	Don‟t change the occurrence, but look for the other q Don‟t change. Leave query replace completely
!	Change this occurrence and all others in the file

RESULT:

Ex.No:2	Programs using the following system calls of UNIX operating system fork, exec, getpid, exit, wait, close, stat, opendir, readdir

AIM:

To write C Programs using the following system calls of UNIX operating system fork, exec, getpid, exit, wait, close, stat, opendir, readdir.

1. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEMS (OPENDIR, READDIR, CLOSEDIR)

ALGORITHM:

STEP 1: Start the program.
STEP 2: Create struct dirent.
STEP 3: declare the variable buff and pointer dptr.
STEP 4: Get the directory name.
STEP 5: Open the directory.
STEP 6: Read the contents in directory and print it.
STEP 7: Close the directory.

PROGRAM:

OUTPUT:

RESULT:

2. PROGRAM FOR SYSTEM CALLS OF UNIX OPERATING SYSTEM (fork, getpid, exit)

ALGORITHM:

STEP 1: Start the program.
STEP 2: Declare the variables pid,pid1,pid2.
STEP 3: Call fork() system call to create process.
STEP 4: If pid==-1, exit.
STEP 5: Ifpid!=-1 , get the process id using getpid().
STEP 6: Print the process id.
STEP 7:Stop the program

PROGRAM:

OUTPUT:

RESULT:

Ex.No:3	C programs to simulate UNIX commands like cp, ls, grep.

AIM:

To write C programs to simulate UNIX commands like cp, ls, grep.

1.Program for simulation of cp unix commands

ALGORITHM:

STEP1: Start the program
STEP 2:Declare the variables ch, *fp, sc=0
STEP3: Open the file in read mode
STEP 4: Get the character
STEP 5: If ch== " " then increment sc value by one
STEP 6: Print no of spaces
STEP 7:Close the file

PROGRAM:

OUTPUT:

2.PROGRAM FOR SIMULATION OF LS UNIX COMMANDS

ALGORTIHM:

STEP1 : Start the program
STEP2 : Open the directory with directory object dp
STEP3 : Read the directory content and print it.
STEP4: Close the directory.

PROGRAM:

OUTPUT:

3. PROGRAM FOR SIMULATION OF GREP UNIX COMMANDS

ALGORITHM

STEP1: Start the program
STEP2: Declare the variables fline[max], count=0, occurrences=0 and pointers *fp, *newline**.**
STEP 3: Open the file in read mode.
STEP4: In while loop check fgets(fline,max,fp)!=NULL
STEP 5: Increment count value.
STEP 6: Check newline=strchr(fline, „\n‟)
STEP 7: print the count,fline value and increment the occurrence value.
STEP 8: Stop the program

PROGRAM:

OUTPUT

RESULT:

Ex.No:4	SIMPLE SHELL PROGRAMS

AIM:

To write simple shell programs by using conditional, branching and looping statements.

1.Write a Shell program to check the given number is even or odd

ALGORITHM:

SEPT 1: Start the program.
STEP 2: Read the value of n.
STEP 3: Calculate „r=expr $n%2".
STEP 4: If the value of r equals 0 then print the number is even
STEP 5: If the value of r not equal to 0 then print the number is odd.

PROGRAM:

OUTPUT

2.Write a Shell program to check the given year is leap year or not

ALGORITHM:

SEPT 1: Start the program.
STEP 2: Read the value of year.
STEP 3: Calculate „b=expr $y%4".
STEP 4: If the value of b equals 0 then print the year is a leap year
STEP 5: If the value of r not equal to 0 then print the year is not a leap year.

PROGRAM:

OUTPUT

3.Write a Shell program to find the factorial of a number

ALGORITHM:

SEPT 1: Start the program.
STEP 2: Read the value of n.
STEP 3: Calculate „i=expr $n-1‟.
STEP 4: If the value of i is greater than 1 then calculate „n=expr $n * $i‟ and „i=expr $i – 1‟
STEP 5: Print the factorial of the given number.

PROGRAM:

OUTPUT

4.Write a Shell program to swap the two integers

ALGORITHM:

SEPT 1: Start the program.
STEP 2: Read the value of a,b.
STEP 3: Calculate the swapping of two values by using a temporary variable temp.
STEP 4: Print the value of a and b.

PROGRAM:

OUTPUT

RESULT:

Ex.No:5	CPU SCHEDULING ALGORITHMS
	PRIORITY

AIM:

To write a C program for implementation of Priority scheduling algorithms.

ALGORITHM:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer, totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign p and allocate the memory.
Step 4: Inside the for loop get the value of burst time and priority.
Step 5: Assign wtime as zero .
Step 6: Check p[i].pri is greater than p[j].pri .
Step 7: Calculate the total of burst time and waiting time and assign as turnaround time.
Step 8: Stop the program.

PROGRAM:

OUTPUT:

RESULT

Ex.No:5.b	CPU SCHEDULING ALGORITHMS
	ROUND ROBIN SCHEDULING

AIM:

To write a C program for implementation of Round Robin scheduling algorithms.

ALGORITHM:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer, totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign p and allocate the memory.
Step 4: Inside the for loop get the value of burst time and priority and read the time quantum.
Step 5: Assign wtime as zero.
Step 6: Check p[i].pri is greater than p[j].pri .
Step 7: Calculate the total of burst time and waiting time and assign as turnaround time.
Step 8: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:5.c	CPU SCHEDULING ALGORITHMS
	FCFS

AIM:

To write a C program for implementation of FCFS and SJF scheduling algorithms.

ALGORITHM:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign pid as I and get the value of p[i].btime.
Step 4: Assign p[0] wtime as zero and tot time as btime and inside the loop calculate wait time and turnaround time.
Step 5: Calculate total wait time and total turnaround time by dividing by total number of process.
Step 6: Print total wait time and total turnaround time.
Step 7: Stop the program.

PROGRAM:

OUTPUT:

RESULT

Ex.No:5.d	CPU SCHEDULING ALGORITHMS
	SJF SCHEDULING

AIM:

To write a C program for implementation of SJF scheduling algorithms.

ALGORITHM:

Step 1: Inside the structure declare the variables.
Step 2: Declare the variable i,j as integer,totwtime and totttime is equal to zero.
Step 3: Get the value of „n" assign pid as I and get the value of p[i].btime.
Step 4: Assign p[0] wtime as zero and tot time as btime and inside the loop calculate wait time and turnaround time.
Step 5: Calculate total wait time and total turnaround time by dividing by total number of process.
Step 6: Print total wait time and total turnaround time.
Step 7: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:6	PRODUCER CONSUMER PROBLEM USING SEMAPHORES

AIM:

To write a C-program to implement the producer – consumer problem using semaphores.

ALGORITHM:

Step 1: Start the program.
Step 2: Declare the required variables.
Step 3: Initialize the buffer size and get maximum item you want to produce.
Step 4: Get the option, which you want to do either producer, consumer or exit from the operation.
Step 5: If you select the producer, check the buffer size if it is full the producer should not produce the item or otherwise produce the item and increase the value buffer size.
Step 6: If you select the consumer, check the buffer size if it is empty the consumer should not consume the item or otherwise consume the item and decrease the value of buffer size.
Step 7: If you select exit come out of the program.
Step 8: Stop the program.

PROGRAM:

OUTPUT:-

RESULT:

Ex.No:7	IPC USING SHARED MEMORY

AIM:

To write a c program to implement IPC using shared memory.

ALGORITHM:

Step 1: Start the process
Step 2: Declare the segment size
Step 3: Create the shared memory
Step 4: Read the data from the shared memory
Step 5: Write the data to the shared memory
Step 6: Edit the data
Step 7: Stop the process

PROGRAM:

OUTPUT:

RESULT:

Ex.No:8	BANKERS ALGORITHM FOR DEADLOCK AVOIDANCE

AIM:

To write a C program to implement banker"s algorithm for deadlock avoidance.

ALGORITHM:

Step-1: Start the program.
Step-2: Declare the memory for the process.
Step-3: Read the number of process, resources, allocation matrix and available matrix.
Step-4: Compare each and every process using the banker"s algorithm.
Step-5: If the process is in safe state then it is a not a deadlock process otherwise it is a deadlock process
Step-6: produce the result of state of process
Step-7: Stop the program

PROGRAM:

OUTPUT:

RESULT:

Ex.No:9	ALGORITHM FOR DEADLOCK DETECTION

AIM:

To write a C program to implement algorithm for deadlock detection.

ALGORITHM:

Step-1: Start the program.
Step-2: Declare the memory for the process.
Step-3: Read the number of process, resources, allocation matrix and available matrix.
Step-4: Compare each and every process using the banker"s algorithm.
Step-5: If the process is in safe state then it is a not a deadlock process otherwise it is a deadlock process
Step-6: produce the result of state of process
Step-7: Stop the program

PROGRAM:

OUTPUT:

RESULT:

Ex.No:10	THREADING & SYNCHRONIZATION APPLICATIONS

AIM:

To write a c program to implement Threading and Synchronization Applications.

ALGORITHM:

Step 1: Start the process
Step 2: Declare process thread, thread-id.
Step 3: Read the process thread and thread state.
Step 4: Check the process thread equals to thread-id by using if condition.
Step 5: Check the error state of the thread.
Step 6: Display the completed thread process.
Step 7: Stop the process

PROGRAM:

OUTPUT:

RESULT:

Ex.No:11.a	MEMORY ALLOCATION METHODS FOR FIXED PARTITION
	FIRST FIT

AIM:

To write a C program for implementation memory allocation methods for fixed partition using first fit.

ALGORITHM:

Step 1:Define the max as 25.
Step 2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max].
Step 3: Get the number of blocks,files,size of the blocks using for loop.
Step 4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
Step 5: Check highest<temp,if so assign ff[i]=j,highest=temp
Step 6: Assign frag[i]=highest, bf[ff[i]]=1,highest=0
Step 7: Repeat step 4 to step 6.
Step 8: Print file no,size,block no,size and fragment.
Step 9: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:11.b	MEMORY ALLOCATION METHODS FOR FIXED PARTITION
	WORST FIT

AIM:

To write a C program for implementation of FCFS and SJF scheduling algorithms.

ALGORITHM:

Step 1:Define the max as 25.
Step 2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max].
Step 3: Get the number of blocks,files,size of the blocks using for loop.
Step 4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
Step 5: Check temp>=0,if so assign ff[i]=j break the for loop.
Step 6: Assign frag[i]=temp,bf[ff[i]]=1;
Step 7: Repeat step 4 to step 6.
Step 8: Print file no,size,block no,size and fragment.
Step 9: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:11.c	MEMORY ALLOCATION METHODS FOR FIXED PARTITION
	BEST FIT

AIM:

To write a C program for implementation of FCFS and SJF scheduling algorithms.

ALGORITHM:

Step 1:Define the max as 25.
Step 2: Declare the variable frag[max],b[max],f[max],i,j,nb,nf,temp, highest=0, bf[max],ff[max].
Step 3: Get the number of blocks,files,size of the blocks using for loop.
Step 4: In for loop check bf[j]!=1, if so temp=b[j]-f[i]
Step 5: Check lowest>temp,if so assign ff[i]=j,highest=temp
Step 6: Assign frag[i]=lowest, bf[ff[i]]=1,lowest=10000
Step 7: Repeat step 4 to step 6.
Step 8: Print file no,size,block no,size and fragment.
Step 9: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:12	PAGING TECHNIQUE OF MEMORY MANAGEMENT

AIM:

To write a c program to implement Paging technique for memory management.

ALGORITHM:

Step 1: Start the process
Step 2: Declare page number, page table, frame number and process size.
Step 3: Read the process size, total number of pages
Step 4: Read the relative address
Step 5: Calculate the physical address
Step 6: Display the address
Step 7: Stop the process

PROGRAM:

OUTPUT:

RESULT:

Ex.No:13.a	PAGE REPLACEMENT ALGORITHMS
	FIFO

AIM:

To write a C program for implementation of FIFO page replacement algorithm.

ALGORITHM:

Step 1: Start the program.
Step 2: Declare the necessary variables.
Step 3: Enter the number of frames.
Step 4: Enter the reference string end with zero.
Step 5: FIFO page replacement selects the page that has been in memory the longest time and when the page must be replaced the oldest page is chosen.
Step 6: When a page is brought into memory, it is inserted at the tail of the queue.
Step 7: Initially all the three frames are empty.
Step 8: The page fault range increases as the no of allocated frames also increases.
Step 9: Print the total number of page faults.
Step 10: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:13.b	PAGE REPLACEMENT ALGORITHMS
	LRU

AIM:

To write a c program to implement LRU page replacement algorithm.

ALGORITHM:

Step 1: Start the process
Step 2: Declare the size
Step 3: Get the number of pages to be inserted
Step 4: Get the value
Step 5: Declare counter and stack
Step 6: Select the least recently used page by counter value
Step 7: Stack them according the selection.
Step 8: Display the values
Step 9: Stop the process

PROGRAM:

OUTPUT:

RESULT:

Ex.No:13.c	PAGE REPLACEMENT ALGORITHMS
	LFU

AIM:

To write C program to implement LFU page replacement algorithm.

ALGORITHM:

Step 1: Start the process
Step 2: Declare the size
Step 3: Get the number of pages to be inserted
Step 4: Get the value
Step 5: Declare counter and stack
Step 6: Select the least frequently used page by counter value
Step 7: Stack them according the selection.
Step 8: Display the values
Step 9: Stop the process

PROGRAM:

OUTPUT:

RESULT:

Ex.No:14.a	FILE ORGANIZATION TECHNIQUE
	SINGLE LEVEL DIRECTORY

AIM:

To write C program to organize the file using single level directory.

ALGORITHM:

Step-1: Start the program.
Step-2: Declare the count, file name, graphical interface.
Step-3: Read the number of files
Step-4: Read the file name
Step-5: Declare the root directory
Step-6: Using the file eclipse function define the files in a single level
Step-7: Display the files
Step-8: Stop the program

FLOWCHART:

PROGRAM:

OUTPUT:

RESULT:

Ex.No:14.b	FILE ORGANIZATION TECHNIQUE
	TWO LEVEL DIRECTORY

AIM:

To write C program to organize the file using two level directory.

ALGORITHM:

Step-1: Start the program.
Step-2: Declare the count, file name, graphical interface.
Step-3: Read the number of files
Step-4: Read the file name
Step-5: Declare the root directory
Step-6: Using the file eclipse function define the files in a single level
Step-7: Display the files
Step-8: Stop the program

PROGRAM:

OUTPUT:

RESULT:

Ex.No:15.a	FILE ALLOCATION STRATEGIES
	SEQUENTIAL

AIM:

To write a C program for sequential file for processing the student information.

ALGORITHM:

Step-1: Start the program.
Step-2: Get the number of records user want to store in the system.
Step-3: Using Standard Library function open the file to write the data into the file.
Step-4: Store the entered information in the system.
Step-5: Using do..While statement and switch case to create the options such as
1-DISPLAY, 2.SEARCH, 3.EXIT.
Step-6: Close the file using fclose() function.
Step-7: Process it and display the result.
Step-8: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:15.b	FILE ALLOCATION STRATEGIES
	LINKED

AIM:

To write a C program for random access file for processing the employee details.

ALGORITHM:

Step-1: Start the program.
Step-2: Get the number of records user want to store in the system.
Step-3: Using Standard Library function open the file to write the data into the file.
Step-4: Store the entered information in the system.
Step-5: Using do..While statement and switch case to create the options such as
1-DISPLAY, 2.SEARCH, 3.EXIT.
Step-6: Close the file using fclose() function.
Step-7: Process it and display the result.
Step-8: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

Ex.No:15.c	FILE ALLOCATION STRATEGIES
	INDEXED

AIM:

To write a C program for random access file for processing the employee details.

ALGORITHM:

Step-1: Start the program.
Step-2: Get the number of records user want to store in the system.
Step-3: Using Standard Library function open the file to write the data into the file.
Step-4: Store the entered information in the system.
Step-5: Using do..While statement and switch case to create the options such as 1-DISPLAY, 2.SEARCH, 3.EXIT.
Step-6: Close the file using fclose() function.
Step-7: Process it and display the result.
Step-8: Stop the program.

PROGRAM:

OUTPUT:

RESULT:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS8481– DATABASE MANAGEMENT SYSTEMS LABORATORY

R 2017
LAB MANUAL

INDEX

Ex.No:1 Data Definition Commands, Data Manipulation Commands for Inserting, Deleting, Updating And Retrieving Tables And Transaction Control Statements

DATA DEFINITION LANGUAGE

CREATE TABLE:

SQL> Create table library_branch (branch_id integer, branch_name varchar2 (50), address varchar2 (50));

Table created.

DESCRIBING THE TABLE CREATED :

SQL> Desc library_branch

CHANGING NAME OF AN OBJECT:

RENAME:

SQL> rename library_branch to library;

Table renamed.

ALTER TABLE:

ADD:
SQL> alter table library add phno number(10);

Table altered.

MODIFY:

SQL> alter table library modify branch_id number(30);

Table altered.

DROP TABLE:

SQL> drop table library;

Table dropped.

DML COMMAND

SQL> create table publisher (name varchar2(20), phone integer, address varchar2 (20));
Table created.

INSERT SQL command

SQL> insert into publisher values ('mcgraw-hill', 9989076587, 'bangalore');

1 row created.

SQL SELECT Statement

SQL> select * from publisher;

SQL UPDATE Statement

SQL> update publisher set name='random house' where phone=9989076587 ;

1 row updated.

Updating Multiple Columns

SQL> update publisher set name='hachette livre' , address='chennai' where phone=9889076565;

1 row updated.

SQL Delete Statement

SQL> delete from publisher where phone=9889076565;

1 row deleted.

SQL TRUNCATE Statement

SQL> TRUNCATE TABLE publisher;

Table truncated.

TRANSACTION_CONTROL_LANGUAGE

COMMIT:

SQL> commit;

Commit complete.

SAVEPOINT:

SQL> savepoint p2;

Savepoint created.

ROLLBACK:

SQL> roll back p2;

Rollback complete.

RESULT:

Ex.no.2 Database Querying –Simple queries, Nested queries, Sub queries and Joins

Simple queries:

SQL> insert into publisher values('&name','&phone','&address');
Enter value for name: mcgraw_hill
Enter value for phone: 9989076587
Enter value for address: bangalore
old 1: insert into publisher values('&name','&phone','&address')
new 1: insert into publisher values('mcgraw_hill', 9989076587, bangalore)

1 row created.

Nested queries:

SQL> create table library_branch (branch_id integer, branch_name varchar2 (50));

Table created.

IN:

SQL> select phone from details where branch_id in (select branch_id from library_branch where branch_name= 'ansit ');

NOT IN:

SQL> select phone from details where branch_id not in (select branch_id from library_branch where branch_name= 'ansit ');

Sub queries:

SQL> select*from library_branch where branch_id in (select branch_id from library_branch where branch_id<=11);

Joins:

1.EQUI-JOIN:-

1.SQL> select*from library_branch;

SQL> create table details(branch_id integer ,phone number(10), address varchar2 (50));

Table created.

SQL> select * from library_branch,details where library_branch.branch_id=details.branch_id;

2.NON-EQUIJOIN:-

SQL> select * from library_branch,details where library_branch.branch_id >=stu.s_id;

3.SELF JOIN:-

SQL> select * from library_branch u,details t where u.branch_id=t.branch_id;

4.INNER JOIN:

SQL> select * from library_branch INNER JOIN details using(branch_id);

5.NATURAL JOIN:

SQL> select * from library_branch NATURAL JOIN details;

CROSS JOIN:

SQL> select * from library_branch CROSS JOIN details;

LEFTOUT-JOIN:-

SQL> select* from library_branch,details where library_branch.branch_id(+)=details. branch _id;

RIGHTOUTER-JOIN:-

SQL> select* from library_branch,details where library_branch.branch_id= details.branch_id(+);

FULLOUTER-JOIN:-

SQL> select* from library_branch FULL JOIN details ON library_branch.branch_id= details.branch_id;

RESULT:

Ex.no.3 Implementation of Views, Synonyms and sequence

CREATING VIEWS:-

SQL> create **view** library as select * from library_branch;
View created.

DISPLAYING THE VIEW TABLE:-

SQL> **select * from library;**

INSERTING THE VALUES VIEW TABLE:-

SQL> **insert into library** values('&branch_id','&branch_name');
Enter value for branch_id: 14
Enter value for branch_name: manipal
old 1: insert into vstu values('&branch_id','&branch_name')
new 1: insert into vstu values(14, 'manipal')
1 row created.

UPDATING THE VIEW TABLE:-

SQL> update library set sname='manipal';
8 rows updated.

DELETING THE ROWS FROM VIEW TABLE:-

SQL> **delete from library** where branch_id=10;
1 row deleted.

DROP THE VIEW TABLE:-

SQL> drop view library;
View dropped.

SEQUENCE

```
SQL> CREATE SEQUENCE seq_1
  2  START WITH 1
  3  INCREMENT BY 1
  4  MAXVALUE 999
  5  CYCLE;
```

Sequence created.

SQL> insert into student_details values('kalai',seq_1.nextval,19);

1 row created.

SYNONYMS

SQL> create synonym student for student_details;
Synonym created.
SQL>drop synonym student;
Synonym dropped;

RESULT:

Ex.No.4 **Implicit and Explicit Cursors**

IMPLICIT CURSOR:

```
SQL> select *from emp;

SQL> SET SERVEROUTPUT ON

SQL>  DECLARE
  2   total_rows number(2);
  3   BEGIN
  4   UPDATE emp
  5   SET sal=sal+5000;
  6   IF sql%notfound THEN
  7   dbms_output.put_line('no employees updated');
  8   ELSIF sql%found THEN
  9   total_rows:=sql%rowcount;
 10   dbms_output.put_line(total_rows||'employees updated');
 11  END IF;
 12  END;
 13  /
4employees updated

PL/SQL procedure successfully completed.
```

EXPLICIT CURSOR:

```
SQL> set serveroutput on

SQL> DECLARE
  2  e_id emp.empno%type;
  3  e_name emp.name%type;
  4  e_job emp.job%type;
  5  CURSOR e_emp is
  6  SELECT empno,name,job from emp;
  7  BEGIN
  8  OPEN e_emp;
  9  LOOP
 10  FETCH e_emp into e_id,e_name,e_job;
```

```
 11  EXIT WHEN e_emp%notfound;
 12  dbms_output.put_line(e_id||' '||e_name||' '||e_job);
 13  END LOOP;
 14  CLOSE e_emp;
 15  END;
 16  /
11 mala programer
12 malar engineer
106 pani manager
15 sara devloper

PL/SQL procedure successfully completed.
```

RESULT:

EX.NO.5 PROCEDURES AND FUNCTIONS

Create Simple Procedures

```
SQL> create or replace procedure hello as
  2  BEGIN
  3  dbms_output.put_line('Hello World');
  4  END;
  5  /
Procedure created.
SQL> execute hello;
Hello World
PL/SQL procedure successfully completed.
```

Procedure Using To Find Minimum Number

```
SQL> create or replace procedure findmin(x in number,y in number,z out number)
  1  is
  6  begin
  7  if x<y then
  8  z:=x;
  9  else
 10  z:=y;
 11  end if;
 12  end;
 13  declare
 14  a number;
 15  b number;
 16  c number;
 17  begin
 18  a:=23;
 19  b:=45;
 20  findmin(a,b,c);
 21  dbms_output.put_line('minimum of'||c);
 22  end;
 23  /
minimum of(23,45)23
PL/SQL procedure successfully completed.
```

Procedure Using Cursor Implementation

```
SQL> create table stu(reg number(3),sname varchar2(5),mark1 number(2),mark2
number(2),mark3 number(2),mark4 number(2),mark5 number(2));

Table created.
SQL> insert into stu values(101,'xxx',99,80,87,88,77);
```

1 row created.

SQL> commit;

Commit complete.

```
SQL>  declare
  2   average number(5,2);
  3   total number(3);
  4   cursor c_mark is select*from stu where mark1>=40 and mark2>=40 and mark3>=40 and
mark4>=40 and mark5>=40;
  5   begin
  6   dbms_output.put_line('reg sname mark1 mark2 mark3 mark4 mark5 total average');
  7   dbms_output.put_line('--------------------------------------------------------------------------');
  8   for stu in c_mark
  9   loop
 10    tot:=stu.mark1+stu.mark2+stu.mark3+stu.mark4+stu.mark5;
 11    average:=total/5;
12dbms_output.put_line(stu.reg||rpad(stu.sname,15)||rpad(stu.mark1,6)||rpad(stu.mark2,6)||rpad(s
tu.mark3,6)||rpad(stu.mark4,6)||rpad(stu.mark5,6)||rpad(total,8)||rpad(average,5));
 13  end loop;
 14  end;
 15  /
```

PL/SQL procedure successfully completed.

Stored Procedures

CREATING A TABLE:-

```
SQL> create table student(roll number(10),mark1 number(10),mark2 number(10),total
number(10));
Table created.
```

INSERTING VALUES TO THE TABLE:-

```
SQL> insert into student values(&roll,&mark1,&mark2,&total);
Enter value for roll: 1
Enter value for mark1: 99
Enter value for mark2: 88
Enter value for total: 0
old   1: insert into student values(&roll,&mark1,&mark2,&total)
new   1: insert into student values(1,99,88,0)
1 row created.
```

PL/SQL PROCEDURE:-

```
SQL> edit pro.sql;
SQL> set serveroutput on
```

```
SQL> get pro.sql;

  1  create or replace procedure student(rnum number)is
  2  m1 number;
  3  m2 number;
  4  tot number;
  5  begin
  6   select mark1,mark2 into m1,m2 from student where roll=rnum;
  7   if m1<m2 then
  8     update student set tot=m1+m2 where roll=rnum;
  9   elsif m1>m2 then
 10     update student set tot=m1+m2 where roll=rnum;
 11   end if;
 12 end;
```

Function

Factorial Program Using Function

```
SQL>declare
          n number;
          fac number:=1;
          i number;
         begin
          n:=&n;
          for i in 1..n
          loop
                    fac:=fac*i;
          end loop;
          dbms_output.put_line('factorial='||fac);
         end;
```

PL/SQL FUNCTIONS:-

```
SQL>  edit fun.sql;
SQL> get fun.sql;
  1  create or replace function student(num number) return number is
  2  tot number;
  3  m1 number;
  4  m2 number;
  5  begin
  6  select mark1,mark2 into m1,m2 from student where roll=num;
  7  tot:=m1+m2;
  8  return tot;
  9* end;
```

RESULT:

EX.NO.6 **IMPLEMENTATION OF TRIGGERS**

Variables used in Triggers

```
SQL> create table customer(id number(3),name varchar2(10),sal number(6));

Table created.

SQL> insert into customer values(101,'Nivin',90000);

1 row created.

SQL> CREATE OR REPLACE TRIGGER display_sal_changes
  2  BEFORE DELETE OR INSERT OR UPDATE ON customer
  3  FOR EACH ROW
  4  WHEN (NEW.ID > 0)
  5  DECLARE
  6  sal_diff number;
  7  BEGIN
  8  sal_diff := :NEW.sal  - :OLD.sal;
  9  dbms_output.put_line('Old sal: ' || :OLD.sal);
 10     dbms_output.put_line('New salary: ' || :NEW.sal);
 11     dbms_output.put_line('Salary difference: ' || sal_diff);
 12  end;
 13  /

Trigger created.

SQL> insert into customer values(104,'Paul',4600);

1 row created.

SQL> update customer set sal=sal+500 where id=102;

1 row updated.
```

Trigger Before Update

```
SQL> create table order(id number(5),quantity number(4),cost number(8,2),date
date,updated_by varchar2(10));

Table created.

SQL> insert into order(id,quantity,cost)values(&id,&quantity,&cost);
Enter value for id: 1
```

```
Enter value for quantity: 4
Enter value for cost: 20
old   1: insert into order(id,quantity,cost)values(&id,&quantity,&cost)
new   1: insert into order(id,quantity,cost)values(1,4,20)

1 row created.

SQL> edit

  1  create or replace trigger order_before_update
  2  before update
  3  on order
  4  for each row
  5  declare
  6  v_username varchar2(10);
  7  begin
  8  select user into v_username from dual;
  9  :new.updated_date:=sysdate;
 10  :new.updated_by:=v_username;
 11* end;
SQL> /

Trigger created.

SQL> update orders set total=3000 where id=2;

1 row updated.
```

Trigger After Update

```
SQL> create table orders(id number(5),quantity number(4),cost number(6,2),total number(8,2));

Table created.

SQL> create table order_audit(id number,quantity_before number,quantity_after number,name
varchar2(20));

Table created.

SQL> insert into orders(id,quantity,cost)values(&id,&quantity,&cost) ;
Enter value for id: 100
Enter value for quantity: 5
Enter value for cost: 10
old   1: insert into orders(id,quantity,cost)values(&id,&quantity,&cost)
new   1: insert into orders(id,quantity,cost)values(100,5,10)
```

```
1 row created.

SQL> edit

Wrote file afiedt.buf

  1  create or replace trigger order_after_update
  2  AFTER UPDATE
  3  ON orders
  4  for each row
  5  declare
  6  v_username varchar2(10);
  7  begin
  8  select user into v_username
  9  from dual;
 10  insert into order_audit(id,quantity_before,quantity_after,username)values(:new.id,
 11* end;
SQL> /

Trigger created.

SQL> update orders set quantity=25 where id=101;

1 row updated.
```

RESULT:

EX.NO.7 EXCEPTION HANDLING IN PL/SQL PROGRAM

EXCEPTION HANDLING USING PL/SQL BLOCK

sql> create table employee(empno number (5),name varchar2(15),address varchar2(35),age number(3));

Table created.

SQL> INSERT INTO EMPLOYEE VALUES(101,'KAVI,'12KKNAGAR'.21);

1 row created.

SQL> SET SERVEROUTPUT ON

```
SQL> DECLARE
  2   C_NO EMPLOYEE.EMPNO%TYPE:=1;
  3  C_NAME EMPLOYEE.NAME%TYPE;
  4  C_ADDR EMPLOYEE.ADDRESS%TYPE;
  5 C_AGE EMPLOYEE.AGE%TYPE:=1;
  6  BEGIN
  7  SELECT NAME,ADDRESS,AGE INTO C_NAME,C_ADDR,C_AGE FROM EMP11;
  8 WHERE EMPNO=C_NO;
  9 DBMS_OUTPUT.PUT_LINE('NAME:'||C_NAME);
10 DBMS_OUTPUT.PUT_LINE('ADDRESS:'||C_ADDR);
11 DBMS_OUTPUT.PUT_LINE('AGE:'||C_AGE);
12 EXCEPTION
13 WHEN NO_DATA_FOUND THEN
14 DBMS_OUTPUT.PUT_LINE('NO SUCH CUSTOMER!');
15 WHEN OTHERS THEN
16 DBMS_OUTPUT.PUT_LINE('ERROR!');
17 END;
18 /
```

OUTPUT:

USER-DEFINED EXCEPTION

SQL> SET SERVEROUTPUT ON

```
SQL> DECLARE
  1   C_NO EMPLOYEE.EMPNO%TYPE:=3;
```

```
 2  C_NAME EMPLOYEE.NAME%TYPE;
 3  C_ADDR EMPLOYEE.ADDRESS%TYPE;
 4  C_AGE EMPLOYEE.AGE%TYPE:=1;
 5  BEGIN
 6  SELECT NAME,ADDRESS,AGE INTO C_NAME,C_ADDR,C_AGE FROM EMP11
 7  WHERE EMPNO=C_NO;
 8  DBMS_OUTPUT.PUT_LINE('NAME:'||C_NAME);
 9  DBMS_OUTPUT.PUT_LINE('ADDRESS:'||C_ADDR);
10  DBMS_OUTPUT.PUT_LINE('AGE:'||C_AGE);
11  EXCEPTION
12  WHEN NO_DATA_FOUND THEN
13  DBMS_OUTPUT.PUT_LINE('NO SUCH CUSTOMER!');
14  WHEN OTHERS THEN
15  DBMS_OUTPUT.PUT_LINE('ERROR!');
16  END;
17  /
```

OUTPUT:

RAISE USER-DEFINED EXCEPTION

```
SQL>  SET SERVEROUTPUT ON

SQL>  DECLARE
 1  C_NO EMPLOYEE.EMPNO%TYPE:=&CC_ID;
 2  C_NAME EMPLOYEE.NAME%TYPE;
 3  C_ADDR EMPLOYEE.ADDRESS%TYPE;
 4  C_AGE EMPLOYEE.AGE%TYPE:=1;
 5  --USER DEFINED EXCEPTION
 6  EX_INVALID_ID EXCEPTION;
 7  BEGIN
 8  IF C_ID<=0 THEN
 9  RAISE EX_INVALID_ID;
10 ELSE
11 SELECT NAME,ADDRESS,AGE INTO C_NAME,C_ADDR,C_AGE FROM EMP11
12  WHERE ID=EMPNO;
13  DBMS_OUTPUT.PUT_LINE('NAME:'||C_NAME);
14  DBMS_OUTPUT.PUT_LINE('ADDRESS:'||C_ADDR);
15 DBMS_OUTPUT.PUT_LINE('AGE:'||C_AGE);
16  END IF;
17  EXCEPTION
18  WHEN EX_INVALID_ID THEN
19  DBMS_OUTPUT.PUT_LINE('ID MUST BE GREATER THAN ZERO!');
20  WHEN NO_DATA_FOUND THEN
21  DBMS_OUTPUT.PUT_LINE('NO SUCH CUSTOMER!');
```

```
22 WHEN OTHERS THEN
23  DBMS_OUTPUT.PUT_LINE('ERROR!');
24  END;
25  /
```

OUTPUT:

RESULT:

EX.NO .08 DATABASE DESIGN USING ER MODELING, NORMALIZATION AND IMPLEMENTATION FOR ANY APPLICATION

Project Title:

Database system for Student Management system

Project Overview Statement

As You Know that a Library is collection of books in any institute .Librarian resposibilty is to manage all the records of books issued and also returned on Manualy.

Current system:

All the Transaction(books issues & books returned) are manualy recorded(registars.)

Students search books by racks it so time consuming
And there is no arrangement.

Also threat of losing recorde.

Project Aim and Objective

The project aim and objective are:

To eliminate the paper –work in library

-to record every transaction in computerized system so that problem such as record file missing won't happen again

Design view

The library has the following tables in its database;

1. student (student_id, studentname, student_email, student_address)

2. Staff (staff_id, staff_name, staff_address, staff_gender, staff_phone)

3. department (department_id, branch_name)

NORAMALIZATION OF TABLE

Why Normalization:

Database normalization is the process of removing redundant data from your tables in order to improve storage efficiency, data integrity, and scalability.

Normalization generally involves splitting existing tables into multiple ones, which must be re-joined or linked each time a query is issued.

Given table is converted to its 1NF as follows.

• STEP NUMBER 1:

elimination of duplicative columns from table 1.

• Step number 2:

create separate table for each group of related data and identify each row with unique column (primary key).

2nd normal form

A table is in first normal form and each non-key field is functionally dependent upon primary key.
Now we'll take the table above and design new tables that will eliminate the repeted date in non key _field

To decide what fields belong together in a table, think about which field

determines the
values in other fields.

Create a table for those fields and enter the sample data.
Think about what the primary key for each table would be and about

the relationship between the tables.
Mark the primary key for each table and make sure that you do not have repeated data in non-key fields.

Third normal form (3NF) requires that there are no functional dependencies of non-key attributes on something other than a candidate key.
A table is in 3NF if all of the non primary-key attributes are mutually independent
There should not be transitive depeinces.

Normalization of Tables in Database

roll_no	name	subject
101	Akon	OS, CN
103	Ckon	Java
102	Bkon	C, C++

After first Normalization:

roll_no	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	Java
102	Bkon	C
102	Bkon	C++

Second normalized Form:

In the following Student relation all attributes are dependent on the primary key StudID

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat

Let's create another table Score, to store the marks obtained by students in the respective subjects. We will also be saving name of the teacher who teaches that subject along with marks.

score_id	student_id	subject_id	marks	teacher
1	10	1	70	Java Teacher
2	10	2	75	C++ Teacher
3	11	1	80	Java Teacher

The simplest solution is to remove columns `teacher` from Score table and add it to the Subject table. Hence, the Subject table will become:

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

And our Score table is now in the second normal form, with no partial dependency.

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

Before third normal form

Staff_id	Name	Gendar	Designation	Address	City	state	cell
1101	Shaid	M	Librarian	House no 12 street 6	Sargodha	punjab	300-1234567
1345	Riaz	m	Data entery	Statilete town	Sargodha	Punjab	0346-1234567
2264	Arshad	m	Naib qaisd	Raza garden	Sargodha	Punjab	0333-1234567

After 3rd Normalization

Student Table

student_id	name	reg_no	branch	address
10	Akon	07-WY	CSE	Kerala
11	Akon	08-WY	IT	Gujarat
12	Bkon	09-WY	IT	Rajasthan

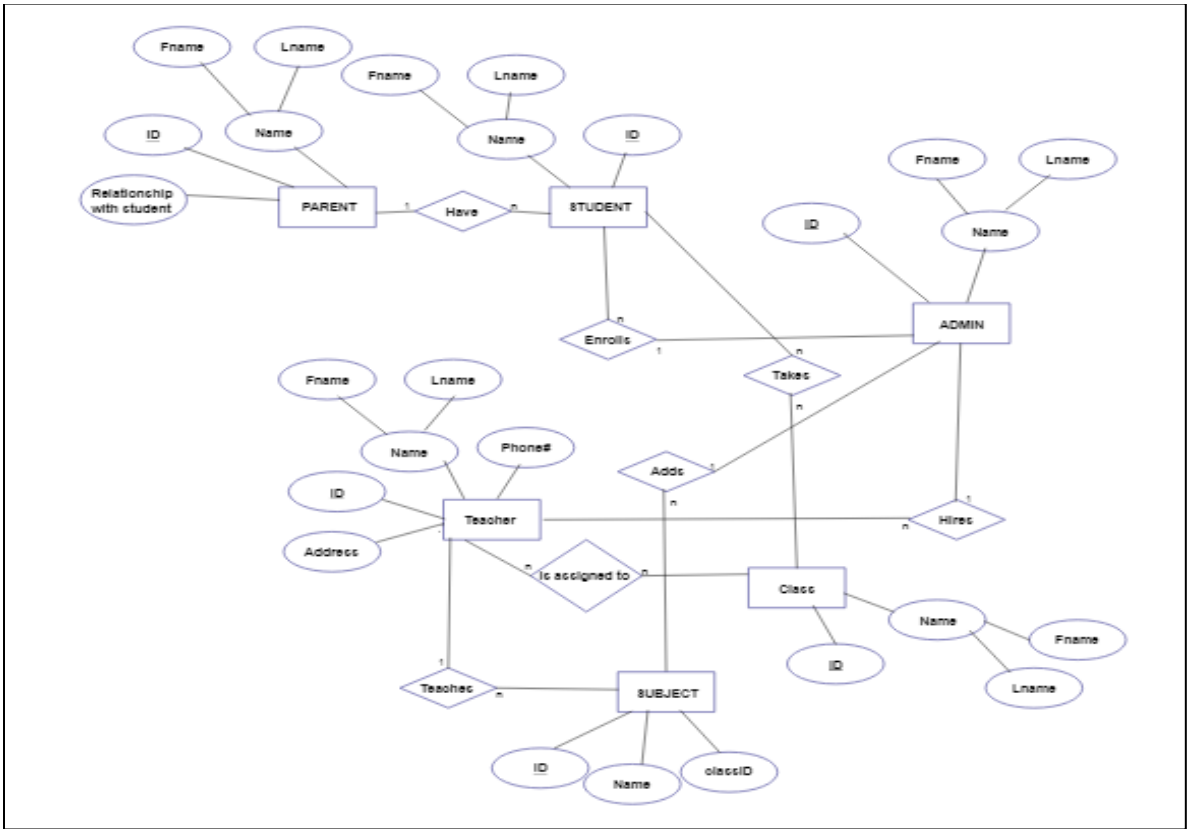
Subject Table

subject_id	subject_name	teacher
1	Java	Java Teacher
2	C++	C++ Teacher
3	Php	Php Teacher

Score Table

score_id	student_id	subject_id	marks
1	10	1	70
2	10	2	75
3	11	1	80

E-R Diagram for School management system



RESULT:

EX.NO .9.

DATABASE CONNECTIVITY WITH FRONT END TOOLS

WEB USER IDENTIFICATION SYSTEM

AIM:

To design the Web User Identification System in visual basic using ORACLE as backend.

CREATING TABLE

SQL> create table webuser(username varchar2(10),password varchar2(10));

Table created.

PROGRAM

RESULT

Ex.No-10. (a)

CASE STUDY USING REAL LIFE DATABASE APPLICATIONS

MATERIAL REQUIREMENT PROCESSING SYSTEM

AIM:

To design the Material Requirement Processing System in visual basic using ORACLE as backend.

CREATING TABLE:

SQL> create table mat(order no varchar2(10),custname varchar2(6),odate date,itemname varchar2(10), itemspec varchar2(10),quantity number(4));

Table created.

PROGRAM 1

OUTPUT:

PROGRAM 2

OUTPUT:

RESULT:

Ex.No-10. (b)

INVENTORY CONTROL SYSTEM

AIM

To design the payroll processing system in visual basic using ORACLE as backend.

CREATING TABLE-Form2

CREATING TABLE-Form3

SQL> create table stockin (ITEMNO NUMBER (4), ITEMNAME VARCHAR2 (10), QUANTITY NUMBER (5), PRICE NUMBER (5));

Table created.

ALGORITHM FOR ADO CONNECTION:

After creating the table in ORACLE, go to start menu

- Start --> Settings --> Control Panel --> Administrative tools --> Data Sources(ODBC) --> User DSN --> Add --> Select ORACLE database driver --> OK.

One new window will appear. In that window, type data source name as table name created in ORACLE. Type user name as secondcsea.

ALGORITHM FOR ADODC IN VISUAL BASIC:

In visual basic create tables, command buttons and then text boxes.

In visual basic, go to start menu.

- Projects --> Components --> Microsoft ADO Data Control 6.0 for OLEDB --> OK.
- Now ADODC Data Control available in tool box.
- Drag and drop the ADODC Data Control in the form.
- Right click in ADODC Data Control, then click ADODC properties.
- One new window will appear.
- Choose general tab, select ODBC Data Sources name as the table name created in ORACLE

- Choose authentication tab and select username password as secondcsea and secondcsea
- Choose record name-->select command type as adcmdTable.
- Select table or store procedure name as table created in ORACLE.
- Click Apply-->OK
- Set properties of each text box.
- Select the data source as ADODC1.
- Select the Data field and set the required field name created in table .

PROGRAM:

OUTPUT:

RESULT:

Ex.No-10.(c)

HOSPITAL MANAGEMENT SYSTEM

AIM

To design the Hospital management System in visual basic using ORACLE as backend.

CREATING TABLE

SQL> create table hos(patname varchar2(10),gender varchar2(6),age number(3),dop date,address varchar2(10),disease varchar2(10));
Table created.

PROGRAM

OUTPUT

RESULT

Ex.No-10.d

TIME TABLE MANAGEMENT SYSTEM

AIM

To design the Time Table Management System in visual basic using ORACLE as backend.

CREATING TABLE

SQL> create table ttable(days varchar2(10),period1 varchar2(15),period2 varchar2(10),period3 varchar2(12),period4 varchar2(12),period5 varchar2(10),period6 varchar2(10));

Table created.

OUTPUT

RESULT

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS3361– DATA SCIENCE LABORATORY

R2021

LAB MANUAL

INDEX

S.No.	Name of the Experiment	Page No	Date of Experiment	Remarks
1.	Working with Numpy Arrays			
2.	Working with Pandas dataframes			
3.	Basic Plots using Matplotlib			
4.	Frequency distributions			
5.	AVERAGES			
6.	Variability Ages			
7.	Normal Curve			
8.	Correlation and Scatter Plot			
9.	Correlation coefficient			
10.	Regression			
11.	Data Analysis			

Exp.No. 1

Working with Numpy Arrays

Aim:

To understand about working with Numpy arrays

Numpy Arrays:

If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

Install it using this command:

```
C:\Users\Your Name>pip install numpy
```

Once NumPy is installed, import it in your applications by adding the import keyword:

```
import numpy as np
```

Numpy Arrays

NumPy is used to work with arrays. The array object in NumPy is called ndarray.

Importing and creating an array in numpy

```
import numpy as np
#1d Arrays
arr1 = np.array([1, 2, 3, 4, 5])
print(arr1)

#2d Arrays
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2)
#3d Arrays
arr3 = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(arr3)
```

Finding the dimensions of a numpy array

```
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])

print(a.ndim)
print(b.ndim)
```

Array indexing

Array indexing is the same as accessing an array element.

```
import numpy as np
#1D arrays
arr = np.array([1, 2, 3, 4])
print(arr[0])


#MultiDimensional Arrays
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('5th element on 2nd row: ', arr[1, 4])
```

Array Slicing

Slicing in python means taking elements from one given index to another given index.

We pass a slice instead of an index like this: [*start*:*end*].

We can also define the step, like this: [*start*:*end*:*step*]

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
#prints from 2nd to 5th element
print(arr[1:5])
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
#print 2nd to 4th element from the 2nd element
print(arr[1, 1:4])
```

Array Shape

NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

```
import numpy as np
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

Array Reshape

By reshaping we can add or remove dimensions or change the number of elements in each dimension.

```
#Converting a 1d array to 2d
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newark = array.reshape(4, 3)
print(newarr)
```

Array Iteration

Iterating means going through elements one by one.

```
#iterating in a 1d array
import numpy as np
arr = np.array([1, 2, 3])
for x in arr:
  print(x)
```

Array joining

Joining means putting contents of two or more arrays in a single array.

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
```

Array Splitting

Splitting is the reverse operation of Joining. Splitting breaks one array into multiple.

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr)
```

Array Sorting

Sorting means putting elements in an *ordered sequence*.

```
import numpy as np
#sorting numbers in ascending order
arr = np.array([3, 2, 0, 1])
print(np.sort(arr))
#sorting in alphabetical order
arr = np.array(['banana', 'cherry', 'apple'])
print(np.sort(arr))
```

Result:

Exp.No. 2

Working with Pandas dataframes

Aim:

To understand about Working with Pandas dataframes

Installation of Pandas

If you have Python and PIP already installed on a system, then installation of Pandas is very easy. It can be installed with this command

```
C:\Users\Your Name>pip install pandas
```

If this command fails, then use a python distribution that already has Pandas installed like Anaconda, Spyder etc.

Importing Pandas

Once Pandas is installed, import it in your applications by adding the import keyword:

```
import pandas as pd
```

Pandas DataFrame:

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Locate Row:

Pandas uses the loc attribute to return one or more specified row(s)

```
#refer to the row index:
print(df.loc[0])
```

Locate Named Indexes

Use the named index in the loc attribute to return the specified row(s)**.**

```
#refer to the named index:
print(df.loc["day2"])
```

Load Files Into a DataFrame

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

Result:

Exp.No. 3

Basic Plots using Matplotlib

Aim:

To understand about basic plots using matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Installation of Matplotlib

If you have Python and PIP already installed on a system, then installation of Matplotlib is very easy.

Install it using this command:

Importing Matplotlib

Once Matplotlib is installed, import it in your applications by adding the import *module* statement:

```
import matplotlib
```

Now Matplotlib is imported and ready to use:

Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

```
import matplotlib.pyplot as plt
```

Now the Pyplot package can be referred to as plt.

Draw a line in a diagram from position (0,0) to position (6,250):

```
import matplotlib.pyplot as plt
```

```
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```

Matplotlib plotting:-

Plotting x and y points

The plot() function is used to draw points (markers) in a diagram.

Draw a line in a diagram from position (1, 3) to position (8, 10):

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

Plotting Without Line

To plot only the markers, you can use the shortcut *string notation* parameter 'o', which means 'rings'.

Draw two points in the diagram, one at position (1, 3) and one in position (8, 10):

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints, 'o')
plt.show()
```

Multiple Points

You can plot as many points as you like, just make sure you have the same number of points in both axes.

Draw a line in a diagram from position (1, 3) to (2, 8) then to (6, 1) and finally to position (8, 10):

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])
plt.plot(xpoints, ypoints)
plt.show()
```

Default X-Points

If we do not specify the points in the x-axis, they will get the default values 0, 1, 2, 3, (etc. depending on the length of the y-points.

So, if we take the same example as above, and leave out the x-points, the diagram will look like this:

Plotting without x-points:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10, 5, 7])
plt.plot(ypoints)
plt.show()
```

Result:

Exp.No. 4

Frequency distributions

Aim:

To find frequency distribution by using matplotlib.

Frequency distribution:

A frequency distribution is a representation, either in a graphical or tabular format, that displays the number of observations within a given interval.

The interval size depends on the data being analyzed and the goals of the analyst.

The intervals must be mutually exclusive and exhaustive. Frequency distributions are typically used within a statistical context.

Importing required libraries:-

Once Matplotlib, seaborn, pandas is installed by using pip install, import it in your applications by adding the import statement:

```
import matplotlib
import pandas as pd
import seaborn as sns
```

Now the required libraries are imported and ready to use.

Dataset:-

Here we are going to use a dataset which has the quality of a diamont, which varies by it's quality.

To import dataset, pd.read_csv() method is used as follow,

```
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/diamonds.csv')
```

Loading the given dataset and Implementing the frequency distribution & finally plotting the data in a graph

Now we're going to do a frequency distribution for the given data of quality which varies with some interval.

In the below program, after implementation we're going to visualize the data by plotting it in a graph using plt.show().

```
sns.set_style("white")
# Import data
df = pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/diamonds.csv')
x1 = df.loc[df.cut=='Ideal', 'depth']
x2 = df.loc[df.cut=='Fair', 'depth']
x3 = df.loc[df.cut=='Good', 'depth']
# Plot
kwargs = dict(hist_kws={'alpha':.6}, kde_kws={'linewidth':2})
plt.figure(figsize=(10,7), dpi= 80)
sns.distplot(x1, color="dodgerblue", label="Compact", **kwargs)
sns.distplot(x2, color="orange", label="SUV", **kwargs)
sns.distplot(x3, color="deep pink", label="minivan", **kwargs)
plt.xlim(50,75)
plt.legend()
```

Result:

Exp.No: 5

AVERAGES

Aim

To understand about Averages working with python numpy.

Numpy Average

To find the average of an numpy array, you can use the numpy.average() function.

Syntax

```
numpy.average(a, axis=None, weights=None, returned=False)
```

Numpy Average

```
# Using numpy to calculate the average of a list
from numpy import mean
numbers = [1,2,3,4,5,6,7,8,9]
average = mean(numbers)
print(average)
```

Numpy Mean

The **numpy mean** function is used for computing the arithmetic mean of the input values.

Syntax

```
numpy.mean(a, axis=some_value, dtype=some_value, out=some_value, keepdims=some_value)
```

Example

```
from numpy import mean
number_list = [45, 34, 10, 36, 12, 6, 80]
avg = mean(number_list)
print("The average is ", round(avg,2))
```

Numpy Median

The numpy median function helps in finding the middle value of a sorted array.

Syntax

```
numpy.median(a, axis=None, out=None, overwrite_input=False, keepdims=False)
```

Example

```
import numpy
speed = [99,86,87,88,86,103,87,94,78,77,85,86]
x = numpy.median(speed)
print(x)
```

Numpy Mode

The Mode value is the value that appears the most number of times.

Syntax

```
scipy.stats.mode(a, axis=0, nan_policy='propagate')
```

Example

Use the SciPy mode() method to find the number that appears the most

```
from scipy import stats
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
x = stats.mode(speed)
print(x)
```

Result

Exp. No:6

Variability Ages

Aim:

To understand about Variability Ages using Pandas

Variability Ages:

Calculating variability of data using pandas

```
#getting data
import pandas as pd
lst = [33219, 36254, 38801, 46335, 46840, 47596, 55130, 56863, 78070, 88830]
sample = pd.Series(lst)
print(type(sample))
sample.mean()
sample.median()
# standard deviation
# default ddof = 1
# divided by n - 1
sample.std()
# standard deviation
# ddof = 0
# divided by n
sample.std(ddof=0
# variance with ddof = 0
# sum((x_i - x_mean)^2) / n
sample.var(ddof=0)
# variance with ddof = 1
# sum((x_i - x_mean)^2) / (n-1)
sample.var(ddof=1)
# mean (average) absolute deviation
Sample.mad()
Summary
lst2 = [38946, 43420, 49191, 50430, 50557, 52580, 53595, 54135, 60181, 62076]sample2 =
pd.Series(lst2)
print(sample 2.std(ddof=0))
print(sample2.mean())
print(sample 2.mad())
```

```
path = './salary.csv'
salary = pd.read_csv(path)


# data read into pandas series
salary.head()

# standard deviation
# degree of freedom = 0
# divided by n instead of divided by n - 1
salary.std(ddof=0)
```

Result:

Exp.No. 7

Normal Curve

Aim:

To understand Normal curves using numpy and matplotlib.

Normal curve

Normal Curve Distribution is a probability function used in statistics that tells about how the data values are distributed. It is the most important probability distribution function used in statistics because of its advantages in real case scenarios. For example, the height of the population, shoe size, IQ level, rolling a die, and many more.

$$z = \frac{x - \mu}{\sigma}$$

Where, x is the variable, mu is the mean, and sigma standard deviation.

Program

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm
import statistics

# Plot between -10 and 10 with .001 steps.
x_axis = np.arange(-20, 20, 0.01)
# Calculating mean and standard deviation
mean = statistics.mean(x_axis)
sd = statistics.stdev(x_axis)
plt.plot(x_axis, norm.pdf(x_axis, mean, sd))
plt.show()
```

Result:

Exp.No. 8

Correlation and Scatter Plot

Aim:

To understand about correlation and scatter plots using matplotlib

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.

Matplotlib is mostly written in python, a few segments are written in C, Objective-C and Javascript for Platform compatibility.

Importing Matplotlib

Once Matplotlib is installed, import it in your applications by adding the import *module* statement:

```
import matplotlib
```

Now Matplotlib is imported and ready to use:

Creating Scatter Plots

With Pyplot, you can use the scatter() function to draw a scatter plot.

The scatter() function plots one dot for each observation. It needs two arrays of the same length, one for the values of the x-axis, and one for values on the y-axis:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

Compare Plots:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y, color = 'hot pink')

x = np.array([2,2,8,1,15,8,12,9,7,3,11,4,7,14,12])
y = np.array([100,105,84,105,90,99,90,95,94,100,79,112,91,80,85])
plt.scatter(x, y, color = '#88c999')
```

Correlation :

Correlation means an association, It is a measure of the extent to which two variables are related. If you analyze any two features of a dataset, then you'll find some type of correlation between those two features.

1. Positive Correlation: When two variables increase together and decrease together. They are positively correlated. '1' is a perfect positive correlation.
2. Negative Correlation: When one variable increases and the other variable decreases together and vice-versa. They are negatively correlated.
3. Zero Correlation(No Correlation): When two variables don't seem to be linked at all. '0' is a perfect negative correlation.

Plotting Correlation matrix using Python

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
y = pd.Series([1, 2, 3, 4, 3, 5, 4])
x = pd.Series([1, 2, 3, 4, 5, 6, 7])
correlation = y.corr(x)
# print the correlation value
correlation
# plotting the data
plt.scatter(x, y)

# This will fit the best line into the graph
plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1)),
        (np.unique(x)), color='red')
# Labelling axes
plt.xlabel('x axis')
plt.ylabel('y axis')
```

Result:

Exp.No. 9

Correlation coefficient

Aim:

To understand the correlation coefficient using pandas.

Correlation

Correlation means an association, It is a measure of the extent to which two variables are related.

If you analyze any two features of a dataset, then you'll find some type of correlation between those two features.

1. Positive Correlation: When two variables increase together and decrease together. They are positively correlated. ‘1’ is a perfect positive correlation.

2. Negative Correlation: When one variable increases and the other variable decreases together and vice-versa. They are negatively correlated.

3. Zero Correlation(No Correlation): When two variables don’t seem to be linked at all. ‘0’ is a perfect negative correlation.

Correlation coefficient using Python

```
import pandas as pd
y = pd.Series([1, 2, 3, 4, 3, 5, 4])
x = pd.Series([1, 2, 3, 4, 5, 6, 7])
correlation = y.corr(x)
# print the correlation value
print(correlation)
```

Result:

Exp.No. 10

Regression

Aim:

To understand about regression using numpy , LinearRegression from sklearn.linear_model.

Regression:.

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable and one or more independent variables .

The most common form of regression analysis is linear regression, in which one finds the line that most closely fits the data according to a specific mathematical criterion .

```
import numpy as np
from sklearn.linear_model import LinearRegression

x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])

model = LinearRegression()
model.fit(x, y)

r_sq = model.score(x, y)

print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)

new_model = LinearRegression().fit(x, y.reshape((-1, 1)))

print('intercept:', new_model.intercept_)
print('slope:', new_model.coef_)
```

```
y_pred = model.predict(x)
print('predicted response:', y_pred, sep='\n')


y_pred = model.intercept_ + model.coef_ * x
print('predicted response:', y_pred, sep='\n')


x_new = np.arange(5).reshape((-1, 1))
print(x_new)


y_new = model.predict(x_new)
print(y_new)
```

Result:

Exp.No. 11

Data Analysis

Aim:

To understand about visualization of data from dataset using numpy , matplotlib , pandas.

Importing modules:

After installing numpy , pandas and matlpotlib modules, import it in your applications by adding the import statement:

```
import numpy
import pandas
import matplotlib
```

Now required modules are imported and ready to use.

Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
population_dict = {'California': 38332521,'Texas': 26448193,'New York': 19651127,'Florida':
19552860,'Illinois': 12882135}
population=pd.Series(population_dict)
area_dict = {'California': 423967, 'Texas': 695662, 'New York': 141297,
'Florida': 170312, 'Illinois': 149995}
area = pd.Series(area_dict)
states = pd.DataFrame({'population': population,
'area': area})
print(states)
plt.bar(states.index,states['population'].values)
plt.title("Population of cities")
plt.show()
```

Result:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS3381– OBJECT ORIENTED PROGRAMMING LABORATORY

R2021

LAB MANUAL

INDEX

S.No	Name of the Experiment	Page No	Date of Experiment	Remarks
1.	**a) Java Program for Sequential Search (Linear Search)**			
	b) Java Program for Binary Search			
	c) Java Program for Selection Sort			
	d) Java Program for Insertion Sort			
2.	a) **Java Application for Stack Data Structure**			
	b) **Java Application for Queue Data Structure**			
3.	**Employee Payroll System**			
4.	**Java Application to Find the Area of different Shapes**			
5.	**Java Application to Find the Area of different Shapes - Interface**			
6.	**Bank Transaction System with User Exceptions**			
7.	**Java Application for Multi threading**			
8.	**Java Application for File Handling**			
9.	**Java Application for Generic Max Finder**			
10.	**Java applications using JavaFX controls, layouts and menus**			

Ex.No: 1 a	Java Program for Sequential Search (Linear Search)
Date:	

Aim:

To create a Java program for Sequential search which is also called as Linear Search.

Algorithm:

Step 1 Start the process.
Step 2 Let the element to be search be **x**.
Step 3 Start from the leftmost element of **arr[]** and one by one compare **x** with each element of **arr[]**.
Step 4 If x matches with an element then return that **index**.
Step 5 If x doesn't match with any of elements then return -1.
Step 6 Exit from the process.
Step 7 Stop the process.

Program:

Output:

Result:

Ex.No: 1 b	Java Program for Binary Search
Date:	

Aim:

To create a Java program for Binary search to find a given element.

Algorithm:

Step 1 Start the process.
Step 2 Compare x with the middle element.
Step 3 If x matches with middle element, we return the mid index.
Step 4 Else If x is greater than the mid element, then x can only lie in the right half subarray after the mid element. So we recur for right half.
Step 5 Else (x is smaller) recur for the left half.
Step 6 Exit from the process.
Step 7 Stop the process.

Program:

Output:

Result:

Ex.No: 1 c	Java Program for Selection Sort
Date:	

Aim:

To create a Java program for Selection sort to sort the given elements.

Algorithm:

Step 1 Start the process.

Step 2 Initialize minimum value(min_idx) to location 0.

Step 3 Traverse the array to find the minimum element in the array.

Step 4 While traversing if any element smaller than min_idx is found then swap both the values.

Step 5 Then, increment min_idx to point to the next element.

Step 6 Repeat until the array is sorted.

Step 7 Stop the process.

Program:

Output:

Result:

Ex.No: 1 d	Java Program for Insertion Sort
Date:	

Aim:

To create a Java program for Insertion sort to sort the given elements.

Algorithm:

Step 1 Start the process.

Step 2 Iterate from arr[1] to arr[N] over the array.

Step 3 Compare the current element (key) to its predecessor.

Step 4 If the key element is smaller than its predecessor, compare it to the elements before.

Step 5 Move the greater elements one position up to make space for the swapped element.

Step 6 Stop the process.

Program:

Output:

Result:

.

Ex.No: 2a	Java Application for Stack Data Structure
Date:	

Aim:

To create a Java console application for stack. Stack operations must be controlled by exception handling techniques.

Algorithm:

Step 1: Start the program
Step 2: Initialize the necessary variables
Step 3: Create a stack, initialize the array and the stack variables
Step 4: Create the push function to insert the elements into the stack.
Step 5: The push function inserts element at the top and displays stack overflow, if the stack is full.
Step 6: Create the pop function to delete the elements from the stack.
Step 7: The pop function deletes the element from the top and displays stack empty, if the stack contains no element.
Step 8: Create isfull() and isempty() to check whether the stack is full or empty.
Step 9: In the main function, perform the operations to insert and delete the elements from the stack.
Step 10: Stop the program.

Program:

Output:

Result:

.

Ex.No: 2b	Java Application for Queue Data Structure
Date:	

Aim:

To create a Java console application for Queue. Queue operations must be controlled by exception handling techniques.

Algorithm:

Step 1: Start the program.
Step 2: Initialize the necessary variables.
Step 3: Initialize the front and rear variables.
Step 4: Create the enqueue function to insert the elements into the queue.
Step 5: The enqueue function inserts element at the rear and displays queue is full, if the queue is full.
Step 6: Create the dequeue function to delete the elements from the queue.
Step 7: The dequeue function deletes the element from the rear and displays queue empty, if the queue contains no element.
Step 8: Create isfull() and isempty() to check whether the queue is full or empty.
Step 9: In the main function, perform the operations to insert and delete the elements from the queue.
Step 10: Stop the program.

Program:

Output:

Result:

Ex.No: 3	Employee Payroll System
Date:	

Aim:

To create a Java console application for employee payroll process management. This project includes Employee and they further classified as Programmer, Assistant Professor, Associate Professor, Professor.

Algorithm:

Step 1 Start the process

Step 2 Prompt the user with converter choice 1. Programmer 2. Assistant Professor 3. Associate Professor 4. Professor 5. Exit and get the choice.

Step 3 If user selects a Programmer then proceed to step 4

Step 4 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 5

Step 5 Proceed to Programmers Payment Processing

- **Step 5.1** Get the basic pay of Programmer
- **Step 5.2** If user enters -1 assume basic pay as 30000 and goto step 15
- **Step 5.3** Else directly go to step 15

Step 6 If user selects Assistant Professor step 7

Step 7 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 8

Step 8 Proceed to Assistant Professor Payment Processing

- **Step 8.1** Get the basic pay of Assistant Professor
- **Step 8.2** If user enters -1 assume basic pay as 25000 and goto step 15
- **Step 8.3** Else directly go to step 15

Step 9 If user selects Associate Professor step 10

Step 10 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 11

Step 11 Proceed to Associate Professor Payment Processing

- **Step 11.1** Get the basic pay of Associate Professor
- **Step 11.2** If user enters -1 assume basic pay as 40000 and goto step 15
- **Step 11.3** Else directly go to step 15

Step 12 If user selects Professor step 13

Step 13 Get the user details [Name, ID, Address, Mail ID and Mobile Number] and goto step 14

Step 14 Proceed to Professor Payment Processing

- **Step 14.1** Get the basic pay of Professor
- **Step 14.2** If user enters -1 assume basic pay as 70000 and goto step 15
- **Step 14.3** Else directly go to step 15

Step 15 Compute Per_Day_Pay = original_basic_pay / no_of_days_in_the_current_month

Step 16 Get the number of days worked from user that include Cl, WH, FH and exclude the LOP

Step 17 Check no_days_worked <= no_of_days_in_the_current_month. Else display “Error Message” and goto step 18

Step 18 Compute Current_Basic_Pay = Per_Day_Pay * no_days_worked

Step 19 Compute Following and Store

DA = (Current_Basic_Pay/100) * 97
HRA = (Current_Basic_Pay/100) * 12
PF = (Current_Basic_Pay/100) * 0.1
GROSS_PAY = Current_Basic_Pay + DA + HRA + PF
NET_PAY = GROSS_PAY - PF

Step 17 Display Payment Details [Name, ID, Address, Mail ID, Mobile Number, BASIC PAY, DA, HRA,PF,GROSS_PAY, NET_PAY].

Step 18 Stop Processing

Program:

Output:

Result:

Ex.No: 4	Java Application to Find the Area of different Shapes
Date:	

Aim:

To create a Java console application to find the area of different shapes using abstract class concept in java.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Rectangle 2. Triangle 3. Circle 4,
Exit Get the choice from user.

Step 3 If user selects Rectangle

Step 3.1 Get the Length and Breath from the user

Step 3.2 Compute Area = Length * Breath

Step 3.3 Display Area .

Step 4 If userselects Triangle

Step 3.1 Get the Length and Height from the user

Step 3.2 Compute Area = Length * Height * 0.5

Step 3.3 Display Area

Step 5 If user selects Circle

Step 5.1 Get the Radius of the Circle

Step 5.2 Compute Area = 3.14 * Radius *

Radius Step 5.3 Display Area

Step 6 If user selects exit end the process

Step 7 Stop the Process.

Program:

Output:

Result:

Ex.No: 5	Java Application to Find the Area of different Shapes - Interface
Date:	

Aim:

To create a Java console application to find the area of different shapes using interface concept in java.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Rectangle 2. Triangle 3. Circle 4, Exit
Get the choice from user.

Step 3 If user selects Rectangle

- Step 3.1 Get the Length and Breath from the user
- Step 3.2 Compute Area = Length * Breath
- Step 3.3 Display Area

Step 4 If user selects Triangle

- Step 3.1 Get the Length and Height from the user
- Step 3.2 Compute Area = Length * Height* 0.5
- Step 3.3 Display Area

Step 5 If user selects Circle

- Step 5.1 Get the Radius of the Circle
- Step 5.2 Compute Area = 3.14 * Radius *Radius
- Step 5.3 Display Area

Step 6 If user selects exit end the process

Step 7 Stop the Process

Program:

Output:

Result:

Ex.No: 6	Bank Transaction System with User Exceptions
Date:	

Aim:

To create a Java console application for Banking transaction system that helps the users to do their credit and debit transactions and it rises user defined exception while encountering errors in transaction and also it should be solved using exception handling techniques.

Algorithm:

Step 1 Start the Process

Step 2 Prompt user with List Operation Choices
1. Add Money 2. Get Money 3. Details 4. Exit
Get the choice from user.

Step 3 If user selects Add Money

- **Step 3.1** Get the amount to be added to balance from the user
- **Step 3.2** If the amount is less than 0 then throw Invalid Credit Exception and goto step 3.4
- **Step 3.3** Else add amount to balance and goto step 2
- **Step 3.4** Prompt user with "Enter valid credit amount" and goto step 3.1

Step 4 If user selects Get Money

- **Step 4.1** Get the amount to be debited to balance from the user
- **Step 4.2** If the amount is greater than existing balance then throw Invalid Debit Exception and goto step 4.4
- **Step 4.3** Else deduct amount from balance and goto step 2
- **Step 4.4** Prompt user with "Enter valid debit amount" and goto step 4.1

Step 5 If user selects Details then display Customer Details [Name, Account Number, Current Balance]

Step 6 If user wants to exit display "Thank You !!!" and end process

Step 7 Stop the Process

Program:

Output:

Result:

Ex.No: 7	Java Application for Multi threading
Date:	

Aim:

To create a Java console application the uses the multi threading concepts in java. The Application has 3 threads one creates random number, one thread computes square of that number and another one computes the cube of that number.

Algorithm:

Step 1 Start the Process

Step 2 Create a thread that generates random number

Step 3 Obtain one random number and check is odd or even

- **Step 3.1** If number is even then create and start thread that computes square of a number
- **Step 3.2** Compute number * number and display the answer
- **Step 3.3** Notify to Random number thread and goto step 4
- **Step 3.4** If number is odd then create and start thread that computes cube of a number
- **Step 3.4** Compute number * number * number and display the answer
- **Step 3.5** Notify to Random number thread and goto step 4

Step 4 Wait for 1 Second and Continue to Step 3 until user wants to exits.

Step 5 Stop the Process

Program:

Output:

Result:

Ex.No: 8	Java Application for File Handling
Date:	

Aim:

To create a Java console application to handle the files and find the file properties [Availability, Readable or Writeable or Both, Length of the File].

Algorithm:

Step 1 Start the Process

Step 2 Prompt the user to enter the file name with path

Step 3 Get the file name

- **Step 3.1** Check the file is exits
- **Step 3.2** If file exists then proceed to step 3.3 else proceed to step 3.8
- **Step 3.3** Check the File is Both Readable and Writeable
- **Step 3.4** If yes display file is “Read and Writeable”
- **Step 3.5** Else check is readable if yes display “Read Only” else move to step 3.6
- **Step 3.6** Else check is writeable if yes display “Write Only”
- **Step 3.7** Compute file size and display
- **Step 3.8** If file not existing then display “File Not Found”

Step 4 Stop the Process

Program:

Output:

Result:

Ex.No: 9	Java Application for Generic Max Finder
Date:	

Aim:

To create a Java console application that finds the maximum in a array based on the type of the elements using generic functions in java.

Algorithm:

Step 1 Start the Process

Step 2 Create a array of number and array of strings and pass it to generic function.

Step 3 If the array is Integer type

- **Step 3.1** Assume first element as MAX
- **Step 3.2** Compare [Numeric Perspetive] this element with MAX
- **Step 3.3** If it is greater than MAX then store current element as MAX
- **Step 3.4** Else do nothing
- **Step 3.5** Goto step 3.1 until all the elements has been processed.

Step 4 If the array is String type

- **Step 4.1** Assume first element as MAX
- **Step 4.2** Compare [Dictionary Perspective] this element with MAX
- **Step 4.3** If it is greater than MAX then store current element as MAX
- **Step 4.4** Else do nothing
- **Step 4.5** Goto step 3.1 until all the elements has been processed.

Step 5 Stop the Process

Program:

Output:

Result:

Ex.No: 10	Java applications using JavaFX controls, layouts and menus
Date:	

Aim:

To create a Java application using JavaFX layouts and Menus to create a menu bar and add a menu to it and also add menu items to menu and also add an event listener to handle the events.

Algorithm:

Step 1: Start the program
Step 2: Import the necessary javafx files.
Step 3: Create the class menubar and set the title for it.
Step 4: Create object for the class and create the necessary menu items in it.
Step 5: Add the menu items using the getitem() and add() functions.
Step 6: Create the label and events for the menu.
Step 7: Create the objects for the necessary classes.
Step 8: In the main function, launch the menu.
Step 9: Stop the program.

Program:

Output:

Result:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS3481– DATABASE MANAGEMENT SYSTEMS LABORATORY

R2021

LAB MANUAL

INDEX

Ex.No : 1 **DDL and DML Commands**
Date :

Aim:

To understand various SQL queries related to creating, altering, and manipulating tables.

Algorithm:

1. Create a table with the required structure using the CREATE TABLE statement.
2. Insert data into the table using the INSERT INTO statement.
3. Alter the table structure using the ALTER TABLE statement.
4. Delete all data from the table using the TRUNCATE TABLE statement.
5. Delete the table using the DROP TABLE statement.
6. Query data from the table using the SELECT statement.
7. Update data in the table using the UPDATE statement.
8. Add constraints to the table using the CONSTRAINT keyword in the CREATE TABLE or ALTER TABLE statement.
9. Describe the structure of the table using the DESC command. For example: DESC table_name;
10. Use the COMMIT statement to save the changes made to the table.

CREATION OF TABLE:

SQL>create table std(sno INT(5),sname varchar(20),age INT(5),sdob date,sm1 INT(4),sm2 INT(4),sm3 INT(4));
Table created.
SQL>insert into std values(101,'AAA',16,'03-jul-88',80,90,98);
1 row created.
SQL>insert into std values(102,'BBB',18,'04-aug-89',88,98,90);

1 row created.

select * from std;

SNO	SNAME	AGE	SDOB	SM1	SM2	SM3
101	AAA	16	03-jul-88	80	90	98
102	BBB	18	04-aug-89	88	98	90

ALTER TABLE WITH ADD:

Sql>create table student(id INT(5),name varchar(10),game varchar(20));
Table created.

Sql>insert into student values(1,'mercy','cricket');
1 row created.

Sql>alter table student add(age INT(4));

Sql>insert into student values(2,'sharmi','tennis',19);

Sql>select * from student;

ID	NAME	GAME
1	Mercy	Cricket

Sql>select * from student;

ID	NAME	GAME	AGE
1	Mercy	cricket	
2	Sharmi	Tennis	19

ALTER TABLE WITH MODIFY:

Sql>alter table student modify id INT(6);

Sql>desc student;

NAME	NULL?	TYPE
Id		INT(6)
Name		Varchar(20)
Game		Varchar(25)
Age		INT(4)

DROP:

Sql>drop table student;
Table dropped.

TRUNCATE TABLE
Sql>Trucate table stud;

DESCRIBE TABLE

SQL>desc emp;

Name	Null?	Type
EmpNo	NOT NULL	number(5)
EName		VarChar(15)
Job	NOT NULL	Char(10)
DeptNo	NOT NULL	number(3)
PHONE_NO		number (10)

DOMAIN INTEGRITY

SQL>Create table cust(custid number(6) not null, name char(10));
SQL>Alter table cust modify (name not null);

CHECK CONSTRAINT

SQL>Create table student (regno number (6), mark number (3) constraint b check (mark >=0 and mark <=100));
SQL>Alter table student add constraint b2 check (length(regno<=4));

ENTITY INTEGRITY

Create a table called EMP with the following structure

Name	Type
EMPNO	NUMBER(6)
ENAME	VARCHAR2(20)
JOB	VARCHAR2(10)
DEPTNO	NUMBER(3)
SAL	NUMBER(7,2)

SQL> create table emp(empno number(6),ename varchar2(20)not null,job varchar2(10) not null, deptno number(3),sal number(7,2));
Table created.

Add a column experience to the emp table. experience numeric null allowed

SQL> alter table emp add(experience number(2));
Table altered.

Modify the column width of the job field of emp table

SQL> alter table emp modify(job varchar2(12));
Table altered.

SQL> alter table emp modify(job varchar(13));
Table altered.

Create dept table with the following structure

Name	Type
DEPTNO	NUMBER(2)
DNAME	VARCHAR2(10)
LOC	VARCHAR2(10)

SQL> create table dept(deptno number(2) primary key,dname varchar2(10),loc varchar2(10));
Table created.

SQL> create table emp1(ename varchar2(10),empno number(6) constraint ch check(empno>100));
Table created.

SQL> alter table emp drop column experience;
Table altered.

SQL> truncate table emp;
Table truncated.

SQL> drop table dept;
Table dropped.

DML COMMANDS

Insert Command

```
Sql>insert into emp values(&empno,'&ename','&job',&deptno,&sal);
Enter value for empno: 1
Enter value for ename: Mathi
Enter value for job: AP
Enter value for deptno: 1
Enter value for sal: 10000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(1,'Mathi','AP',1,10000)
1 row created.
/
Enter value for empno: 2
Enter value for ename: Arjun
Enter value for job: ASP
Enter value for deptno: 2
Enter value for sal: 12000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(2,'Arjun','ASP',2,12000)
1 row created.
 /
Enter value for empno: 3
Enter value for ename: Gugan
Enter value for job: ASP
Enter value for deptno: 1
Enter value for sal: 12000
old 1: insert into emp values(&empno,'&ename','&job',&deptno,&sal)
new 1: insert into emp values(3,'Gugan','ASP',1,12000)
1 row created.
```

Update Command

Sql>select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	12000
3	Gugan	ASP	1	12000

Sql>update emp set sal=15000 where job='ASP'; 2 rows updated.
Sql> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL

1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000

Select Command

Sql>select ename, job from emp;

ENAME	JOB
Mathi	AP
Arjun	ASP
Gugan	ASP
Karthik	Prof
Akalya	AP
suresh	lect

6 rows selected.

Delete Command

Sql>delete from emp where job='lect';

1 row deleted.

Sql>select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000
5	Akalya	AP	1	10000

Orderby

Sql>select * from emp order by sal;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
5	Akalya	AP	1	10000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000
4	Karthik	Prof	2	30000

Orderby (descending order)

Sql>select * from emp order by sal desc;

EMPNO	ENAME	JOB	DEPTNO	SAL
4	Karthik	Prof	2	30000
2	Arjun	ASP	2	15000
3	Gugan	ASP	1	15000

1	Mathi	AP	1	10000
5	Akalya	AP	1	10000

Where

Sql>select * from emp where deptno=1;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
3	Gugan	ASP	1	15000
5	Akalya	AP	1	10000

Distinct

Sql>select distinct deptno from emp;

DEPTNO
1
2

Result:

Ex.No: 2 **REFERENTIAL INTEGRITY**
Date :

Aim:

To demonstrate the usage of various referential integrity.

Algorithm:

1. Create a table named "emp" with columns
2. Create a table named "depositor" with columns
3. Create a table named "depositor1" with columns
4. Create a table named "customer" with columns
5. Create a table named "customer1" with columns
6. Create a table named "loan" with columns
7. Create a table named "loan1" with columns
8. Create a table named "books" with columns to enforce the constraint that the value in this column must exist in the "author" table.
9. Create a table named "books" with columns

NOT NULL:

SQL>create table emp(e_id varchar(5) NOT NULL,e_name varchar(10), e_design varchar(10),dept varchar(10),mgr varchar(10),salary number(10));

UNIQUE :

SQL>create table depositor(customer_name varchar(10),acc_no number(15) UNIQUE, brach_name varchar(10));

SQL>create table depositor1(customer_name varchar(10),acc_no number(15), brach_name varchar(10),UNIQUE(acc_no));

PRIMARY KEY:

SQL>create table customer(customer_id number (5) PRIMARY KEY, customer_name varchar(10),customer_street varchar(10),brach_name varchar(10));

SQL>create table customer1(customer_id number (5),customer_name varchar(10),customer_street varchar(10),brach_name varchar(10),PRIMARY KEY(customer_id));

CHECK:

SQL>create table loan(loan_no varchar(10),customer_name varchar(10), balance number (10) CHECK(balance>1000));

Sql>create table loan1(loan_no varchar(10),customer_name varchar(10), balance number (10), CHECK(balance>1000));

FOREIGN KEY:

Sql>CREATE TABLE books (book_id NUMBER(3), book_title VARCHAR2(30), book_price NUMBER(3), book_author_id NUMBER(3) REFERENCES author(author_id));

Sql>CREATE TABLE books (book_id NUMBER(3) CONSTRAINT bok_bi_pk PRIMARY KEY, book_title VARCHAR2(30), book_price NUMBER(3), book_author_id NUMBER(3), CONSTRAINT bok_ai_fk FOREIGN KEY (book_author_id) REFERENCES author(author_id));

Result:

Ex.No: 3 **'WHERE' CLAUSE AND AGGREGATE FUNCTIONS**
Date:

Aim:

To demonstrate the usage of WHERE clause and aggregative functions in SQL.

Algorithm:

1. To retrieve the ID, Name and Salary of customers whose salary is greater than 2000
2. To retrieve the ID, Name and Salary of a customer with the name Hardik
3. To retrieve the ID, Name and Salary of customers whose salary is greater than 2000 and age is less than 25
4. To retrieve the ID, Name and Salary of customers whose salary is greater than 2000 OR age is less than 25
5. To count the number of employees in the emp table
6. To retrieve the sum of salaries of all employees in the emp table
7. To calculate the average of a given set of values (10, 15, 30)
8. To retrieve the maximum salary of all employees in the emp table
9. To retrieve the department number and the maximum salary of employees in each department where the maximum salary is less than 3000
10. To retrieve the department number and the minimum salary of employees in each department where the minimum salary is greater than 1000

Sql>select * from CUSTOMERS;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
```

```
+----+----------+----------+
```

Fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name Hardik

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE NAME = 'Hardik';
```

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  5 | Hardik   |  8500.00 |
+----+----------+----------+
```

The AND Operator

Sql>Select * from CUSTOMERS;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 and the age is less than 25 years

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 AND age < 25;
```

```
+----+--------+----------+
| ID | NAME   | SALARY   |
+----+--------+----------+
|  6 | Komal  |  4500.00 |
|  7 | Muffy  | 10000.00 |
+----+--------+----------+
```

The OR Operator

Sql>select * from CUSTOMERS;

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

Fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 OR the age is less than 25 years

```
SQL> SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000 OR age < 25;
```

```
+----+----------+----------+
| ID | NAME     | SALARY   |
+----+----------+----------+
|  3 | kaushik  |  2000.00 |
|  4 | Chaitali |  6500.00 |
|  5 | Hardik   |  8500.00 |
|  6 | Komal    |  4500.00 |
|  7 | Muffy    | 10000.00 |
+----+----------+----------+
```

Aggregative functions:

Sql>SELECT COUNT (Sal) FROM emp;

Sql>SELECT SUM (Sal) From emp;

Sql>Select AVG (10, 15, 30) FROM DUAL;

Sql> SELECT MAX (Sal) FROM emp;

SQL> select deptno, max(sal) from emp group by deptno;

SQL> select deptno, max (sal) from emp group by deptno having max(sal)<3000;

```
DEPTNO          MAX(SAL)
--------------------------

30                    2850
```

SQL>select deptno,min(sal) from emp group by deptno having min(sal)>1000;

DEPTNO	MIN (SAL)
10	1300

Result:

Ex.No: 4 **SUB QUERIES AND SIMPLE JOIN OPERATIONS**

Date:

Aim:

To practice subqueries and simple join operations in SQL by working with the given tables.

Algorithm:

1. Create tables "department", "course", "instructor", "section", and "teaches" using the given SQL commands.
2. Write SQL queries to retrieve the required information as per the given questions.
3. Execute the SQL queries and display the results.

Sub queries:

Table 1:

Sql>Create table department (deptname varchar(20), building varchar (15),
budget INT(12,2), primary key(deptname));

Table created.
Sql>Insert into department values('ECE','block1',70000);
Sql>Insert into department values('CSE','block2',70000);
Sql>Insert into department values('EEE','block3',70000);

Table 2:
Sql>create table course (courseid varchar(8) primary key, title varchar(50),
deptname varchar(20), creditsINT(2), foreign key (deptname) references department) ;

Table created.

Sql>Insert into course values('cs101','python','CSE',4);
Sql>Insert into course values('cs102','java','CSE',4);
Sql>Insert into course values('ec102','Electronics circuits','ECE',3);
Insert into course values('ec202','Microprocessor','ECE',3);

Table 3:

Sql>create table instructor (ID varchar(5), name varchar(20) not null,
deptname varchar(20), salary INT(8,2), primary key (ID), foreign key
(deptname) references department);

Sql>insert into instructor values ('1002', 'Sumanth', 'ECE', 66000);
Sql>insert into instructor values ('1001', 'Sumitha', 'CSE', 56000);
Sql>insert into instructor values ('1007', 'Malar', 'CSE', 96000);
Sql>insert into instructor values ('1004', 'Mani', 'ECE', 36000);

Table 4:

Sql>Create table section (courseid varchar(8), secid varchar(8), semester varchar(6), year INT(4), building varchar(15) , roomnumner varchar(7), timeslot varchar(4), **primary key** (*courseid*, *secid*, *semester*, *year*), **foreign key** (*courseid*) **references** *course*);

Sql>Insert into section values('cs101','A','odd',2017,'block2','111','11');
Sql>Insert into section values('cs101','A','even',2018,'block2','222','5');
Sql>Insert into section values('cs102','A','even',2016,'block2','77','8')

Sql>Insert into section values('ec202','A','even',2016,'block1','47','8');
Sql>Insert into section values('ec202','B','odd',2017,'block1','47','8');
Sql>Insert into section values('ec202','C','even',2018,'block1','47','8');

Table 5:

Sql>create table teaches (ID varchar(5), courseid varchar(8), secid varchar(8), semester varchar(6), year INT(4), primary key (ID, courseid, secid, semester, year), foreign key (ID) references instructor, foreign key (courseid, secid, semester, year) references section);

Sql> Insert into teaches values('1001', 'cs101','A','odd',2017);
Sql>Insert into teaches values('1007', 'cs102','A','even',2016);
Sql> Insert into teaches values('1002', 'ec202','A','even',2016);

Display all employee names and salary whose salary is greater than minimum salary of the company and job title starts with 'A'

Sql>select ename,sal from emp where sal>(select min(sal) from emp where job like 'A%');

ENAME	SAL
Arjun	12000
Gugan	20000
Karthik	15000

Find all the employees who work in the same job as Arjun.

Sql>select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	12000
3	Gugan	ASP	2	20000
4	Karthik	AP	1	15000

Sql>select ename from emp where job=(select job from emp where ename='Arjun');

ENAME
Arjun
Gugan

Result:

SIMPLE JOINS

SQL> select * from emp;

EMPNO	ENAME	JOB	DEPTNO	SAL
1	Mathi	AP	1	10000
2	Arjun	ASP	2	12000
3	Gugan	ASP	2	20000
4	Karthik	AP	1	15000

SQL> select * from dept;

DEPTNO	DNAME	LOC
1	ACCOUNTING	NEW YORK
2	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EQUI-JOIN

Display the employee details, departments that the departments are same in both the emp and dept

SQL> select * from emp,dept where emp.deptno=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
1	Mathi	AP	1	10000	1	ACCOUNTING	NEW YORK
2	Arjun	ASP	2	12000	2	RESEARCH	DALLAS
3	Gugan	ASP	2	20000	2	RESEARCH	DALLAS
4	Karthik	AP	1	15000	1	ACCOUNTING	NEW YORK

NON-EQUIJOIN

Display the employee details, departments that the departments are not same in both the emp and dept.

SQL> select * from emp,dept where emp.deptno!=dept.deptno;

EMPNO	ENAME	JOB	DEPTNO	SAL	DEPTNO	DNAME	LOC
2	Arjun	ASP	2	12000	1	ACCOUNTING	NEW YORK
3	Gugan	ASP	2	20000	1	ACCOUNTING	NEW YORK
1	Mathi	AP	1	10000	2	RESEARCH	DALLAS

EMPNO ENAME JOB DEPTNO SAL DEPTNO DNAME LOC

- -

```
4 Karthik      AP     1       15000  2 RESEARCH        DALLAS
1       Mathi AP      1       10000  30 SALES     CHICAGO
2       Arjun  ASP    2       12000  30 SALES     CHICAGO

EMPNO ENAME     JOB    DEPTNO       SAL    DEPTNO DNAME    LOC
----------------------------------------------------------------------


3       Gugan ASP     2       20000  30 SALES     CHICAGO
4       Karthik       AP      1       15000  30 SALES     CHICAGO
1       Mathi AP      1       10000  40 OPERATIONS     BOSTON

EMPNO ENAME     JOB    DEPTNO       SAL    DEPTNO DNAME    LOC
----------------------------------------------------------------------


2        Arjun  ASP    2       12000  40 OPERATIONS     BOSTON
3       Gugan ASP     2       20000  40 OPERATIONS     BOSTON
4       Karthik       AP      1       15000  40 OPERATIONS     BOSTON
12 rows selected.
```

LEFTOUT-JOIN

```
SQL> select * from stud1;

Regno Name  Mark2 Mark3 Result
-------------------------------------


101             john    89      80      pass
102     Raja            70      80      pass
103     Sharin  70              90              pass
104     sam             90              95              pass

SQL> select * from stud2;

 NAME         GRA
-------------------------


john    s
raj     s
sam     a
sharin  a

SQL> select stud1.name,grade from stud1 left outer join stud2 on stud1.name=stud2.name;

Name  Gra
----------------
john    s
raj     s
sam     a
sharin a
```

smith null

RIGHTOUTER-JOIN

Display the Student name, register no, and result by implementing a right outer join

SQL> select stud1.name, regno, result from stud1 right outer join stud2 on stud1.name = stud2.name;

Name Regno Result
---

john 101 pass
raj 102 pass
sam 103 pass
sharin 104 pass

Regno Name Mark1 Mark2 Total
---

1 sindu90 95 185
2 arul 90 90 180

FULLOUTER-JOIN

Display the Student name register no by implementing a full outer join

SQL> select stud1.name, regno from stud1 full outer join stud2 on (stud1.name= stud2.name);

Name Regno
---

john 101
raj 102
sam 103
sharin 104

SELFJOIN

SQL> select distinct ename from emp x, dept y where x.deptno=y.deptno;

ENAME

Arjun
Gugan
Karthik
Mathi

Display the details of those who draw the salary greater than the average salary

SQL> select distinct * from emp x where x.sal >= (select avg(sal) from emp);

EMPNO ENAME JOB DEPTNO SAL
---

3 Gugan	ASP	2	20000	
4 Karthik	AP	1	15000	
11 kavitha	designer		12	17000

Result:

Ex.No: 5 **SET OPERATORS**

Date:

Aim:

The aim of this task is to demonstrate the use of SQL set operators

Algorithm:

1. To display all the department numbers available in both the dept and emp tables without duplicates
2. To display all the department numbers available in both the dept and emp tables
3. To display all the department numbers available in the emp table but not in the dept table
4. To display all the department numbers available in the dept table but not in the emp table

UNION

Display all the dept numbers available with the dept and emp tables avoiding duplicates
Solution:

SQL> select deptno from emp union select deptno from dept;

DEPTNO

1
2
12
30
40

UNION ALL

Display all the dept numbers available with the dept and emp tables.

SQL> select deptno from emp union all select deptno from dept;

DEPTNO

1
2
2
1
12
1
2
30
40
9 rows selected.

MINUS

Display all the dept numbers available in emp and not in dept tables and vice versa

SQL> select deptno from emp minus select deptno from dept;

DEPTNO

12

SQL> select deptno from dept minus select deptno from emp;

DEPTNO

30
40

Result:

Ex.No: 6 **FUNCTIONS AND STORED PROCEDURES**

Date:

Aim:

The aim of the code is to demonstrate the creation and execution of functions and stored procedures **.**

Algorithm:

1. To create a function that returns the total number of employees in a table
2. To execute the function and display the total number of employees
3. To create a stored procedure that displays a welcome message
4. To execute the stored procedure and display the welcome message
5. To create a stored procedure that finds the minimum value of two integers
6. To execute the stored procedure and display the minimum value
7. To create a table and insert data into it

Functions

```
CREATE OR REPLACE FUNCTION totalemployee return INT as
  total INT;
  begin
  SELECT count(*) into total from emp22;
 Return total;
 end;
 /
Function created.
declare
  a INT:=0;
  begin
  a:=totalemployee();
 dbms_output.put_line('Total employees are '||a);
 end;
 /
Total employees are 2
PL/SQL procedure successfully completed.
```

Procedures

Program 1:

```
set serveroutput on;
create or replace procedure p1 as
  begin
  dbms_output.put_line('welcome');
  end;
  /
Procedure created.

execute p1;
```

```
welcome
PL/SQL procedure successfully completed.
```

Program 2:

```
create PROCEDURE findMin(x IN INT, y IN INT, z OUT INT) IS
 BEGIN
 IF x < y THEN
    z:= x;
   ELSE
      z:= y;
   END IF;


END;
 /
Procedure created.

declare
   a INT;
   b INT;
   c INT;
   begin
   a:=23;
  b:=4;
findMin(a,b,c);
  dbms_output.put_line('Minimum value is:'||c);
  end;
   /
Minimum value is:4
PL/SQL procedure successfully completed.
```

Program 3:

```
set serveroutput on;

sql>create table emp22(id INT,name varchar(20),designation varchar(20),salary INT);
Table created.

Sql>insert into emp22 values(3,'john','manager',100000);
1 row created.

Sql>insert into emp22 values(3,'jagan','hr',400000);
1 row created.
```

Result:

Ex.No: 7 TCL AND DCL COMMANDS

Date:

Aim:

To demonstrate the usage of TCL (Transaction Control Language) and DCL (Data Control Language) commands in SQL.

Algorithm:

1. Savepoint:
   a. Create a savepoint named S1 using the SQL command SAVEPOINT S1;
   b. Perform a SELECT query to display the contents of the 'emp' table;
   c. Insert a new row into the 'emp' table using the INSERT INTO command;
   d. Perform another SELECT query to display the contents of the 'emp' table;
   e. Rollback to the savepoint S1 using the ROLLBACK S1 command;
   f. Perform the SELECT query again to confirm that the changes made in step c have been rolled back.
2. Commit:
   a. Perform some SQL operations that modify the data in the database;
   b. Use the COMMIT command to save the changes to the database.
3. Grant:
   a. Develop a SQL query to grant all privileges of the 'employees' table to the 'departments' table using the GRANT ALL command;
   b. Develop another SQL query to grant specific privileges (SELECT, UPDATE, and INSERT) of the 'employees' table to the 'departments' table with the GRANT command.
4. Revoke:
   a. Develop a SQL query to revoke all privileges of the 'employees' table from the 'departments' table using the REVOKE ALL command;
   b. Develop another SQL query to revoke specific privileges (SELECT, UPDATE, and INSERT) of the 'employees' table from the 'departments' table with the REVOKE command.

```
SQL> INSERT INTO EMP VALUES(5,'Akalya','AP',1,10000);
1 row created.

SQL> select * from emp;

EMPNO ENAME JOB DEPTNO SAL
- - - - - - - ---------- - - - - - - - - - - - - - - - - - - - -

1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000
5 Akalya AP 1 10000
```

ROLLBACK

```
rollback s1;
```

select * from emp;

EMPNO ENAME JOB DEPTNO SAL

1 Mathi AP 1 10000
2 Arjun ASP 2 15000
3 Gugan ASP 1 15000
4 Karthik Prof 2 30000

COMMIT

SQL>COMMIT;
Commit complete.

GRANT

Develop a query to grant all privileges of employees table into departments table

SQL> Grant all on employees to departments;

Grant succeeded.

Develop a query to grant some privileges of employees table into departments table

SQL> Grant select, update , insert on departments to departments with grant option;

Grant succeeded.

REVOKE

Develop a query to revoke all privileges of employees table from departments table

SQL> Revoke all on employees from departments;

Revoke succeeded.

Develop a query to revoke some privileges of employees table from departments table

SQL> Revoke select, update , insert on departments from departments;

Revoke succeeded.

Result:

Ex.No: 8 TRIGGERS

Date:

Aim:

To demonstrate the usage of triggers **.**

Algorithm:

1. Create a table "employee22" with columns "empid", "empname", "empdept", "salary".
2. Create a trigger "display_salary" to display the old and new salary values and the difference in salary before any delete or update operation is performed on the "employee22" table.
3. Create a trigger "Check_age" before the insert operation on the "employee" table for each row.
4. Check if the age of the employee being inserted is less than 25.
5. If the age is less than 25, raise an error with the message "ERROR: AGE MUST BE ATLEAST 25 YEARS!" using the SQLSTATE '45000'.
6. End the trigger.

Table creation:

Sql>Create table employee22(empid INT,empname varchar(20),empdept varchar(20),salary INT);

```
CREATE OR REPLACE TRIGGER display_salary
BEFORE DELETE OR UPDATE ON employee22
FOR EACH ROW
DECLARE
  sal_diff INT;
BEGIN
  sal_diff := :NEW.salary - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
```

Trigger to ensure that no employee of age less than 25 can be inserted in the database.

```
CREATE TRIGGER Check_age BEFORE INSERT ON employee
FOR EACH ROW
BEGIN
IF NEW.age < 25 THEN
SIGNAL SQLSTATE '45000'
SET MESSAGE_TEXT = 'ERROR:
      AGE MUST BE ATLEAST 25 YEARS!';
END IF;
END;
/
```

Result:

Ex.No: 9 VIEW AND INDEX

Date:

Aim:

The aim of this exercise is to understand the concepts of views and indexes in SQL

Algorithm:

1. Create a table named "EMPLOYEE" with columns "EMPLOYEE_NAME", "EMPLOYEE_NO", "DEPT_NAME", "DEPT_NO", and "DATE_OF_JOIN".
2. Create a view named "EMPVIEW" using the "CREATE VIEW" statement with columns "EMPLOYEE_NAME", "EMPLOYEE_NO", "DEPT_NAME", "DEPT_NO", and "DATE_OF_JOIN" selected from the "EMPLOYEE" table.
3. Display the view using the "SELECT" statement with the view name.
4. Insert a row into the view using the "INSERT INTO" statement.
5. Drop the view using the "DROP VIEW" statement.
6. Create a view named "EMP_TOTSAL" using the "CREATE VIEW" statement with columns "EMPNO", "ENAME", and "SALARY" selected from the "EMPL" table.
7. Create an index named "idx_lastname" on the "LastName" column of the "Persons" table using the "CREATE INDEX" statement.
8. Alter an index using the "ALTER INDEX" statement by specifying the index name and the table and column(s) to be altered.
9. Drop an index using the "DROP INDEX" statement with the index name specified.

Sql>CREATE TABLE **EMPLOYEE** (EMPLOYEE_NAME VARCHAR(10), EMPLOYEE_NO INT(8), DEPT_NAME VARCHAR(10), DEPT_NO INT (5),DATE_OF_JOIN DATE);

Table created.

CREATION OF VIEW

Sql>CREATE VIEW **EMPVIEW** AS SELECT EMPLOYEE_NAME, EMPLOYEE_NO,DEPT_NAME, DEPT_NO, DATE_OF_JOIN FROM **EMPLOYEE**;

View Created.

DISPLAY VIEW:

SELECT * FROM EMPVIEW;

EMPLOYEE_N	EMPLOYEE_NO	DEPT_NAME	DEPT_NO
RAVI	124	ECE	89
VIJAY	345	CSE	21
RAJ	98	IT	22
GIRI	100	CSE	67

Sql>INSERT INTO EMPVIEW VALUES ('SRI', 120,'CSE', 67);1 ROW CREATED.

Sql>DROP VIEW EMPVIEW;

view dropped

sql>CREATE OR REPLACE VIEW EMP_TOTSAL AS SELECT EMPNO "EID", ENAME "NAME",SALARY "SAL" FROM EMPL;

CREATE INDEX

Sql>CREATE INDEX idx_lastname ON Persons (LastName);

ALTER INDEX

Sql>ALTER INDEX <index name> ON <table name> (<column(s)>);

DROP INDEX

Sql>DROP INDEX *index_name*;

Result:

Ex.No:10 **XML DATABASE**

Date:

Aim:

To provide an introduction to XML (eXtensible Markup Language)

Algorithm:

1. Define XML and its structure.
2. Explain the importance of well-formed and valid XML documents.
3. Describe DTD and its role in defining the legal elements of an XML document.
4. Explain the use of XML schema as an alternative to DTD and how it defines the structure of an XML document.
5. Provide an example of XML schema with its elements and attributes.
6. Describe the different data types available in XML schema, including simpleType and complexType.
7. Discuss the process of XML validation against DTD or schema.
8. Provide a resource for validating XML documents.
9. Conclude the document with a summary of key points.

```
<?xml version="1.0"?>
 <contact-info>
   <contact1>

     <name>Vimal Jaiswal</name>
     <company>SSSIT.org</company>
     <phone>(0120) 4256464</phone>
   </contact1>
   <contact2>
     <name>Mahesh Sharma </name>
     <company>SSSIT.org</company>
     <phone>09990449935</phone>
   </contact2>
 </contact-info>
```

In the above example, a table named contacts is created and holds the contacts (contact1 and contact2). Each one contains 3 entities name, company and phone.

XML Validation

A well formed XML document can be validated against DTD or Schema.

A well-formed XML document is an XML document with correct syntax. It is very necessary to know about valid XML document before knowing XML validation.

Valid XML document

It must be well formed (satisfy all the basic syntax condition)

It should be behave according to predefined DTD or XML schema

XML DTD

A DTD defines the legal elements of an XML document

In simple words we can say that a DTD defines the document structure with a list of legal elements and attributes.

XML schema is a XML based alternative to DTD.

Actually DTD and XML schema both are used to form a well formed XML document.

We should avoid errors in XML documents because they will stop the XML programs.

XML schema

It is defined as an XML language
Uses namespaces to allow for reuses of existing definitions
It supports a large number of built in data types and definition of derived data types

Checking Validation

An XML document is called "well-formed" if it contains the correct syntax. A well-formed and valid XML document is one which have been validated against Schema.
Visit **http://www.xmlvalidation.com** to validate the XML file against schema or DTD.

XML Schema Example

Let's create a schema file.
employee.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">


<xs:element name="employee">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element name="email" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
1.        </xs:element>
2.
3.        </xs:schema>
```

Let's see the xml file using XML schema or XSD file.
employee.xml

```
<?xml version="1.0"?>
<employee
xmlns="http://www.javatpoint.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.javatpoint.com employee.xsd">

  <firstname>vimal</firstname>
  <lastname>jaiswal</lastname>
  <email>vimal@javatpoint.com</email>
</employee>
```

Description of XML Schema

<xs:element name="employee"> : It defines the element name employee.
<xs:complexType> : It defines that the element 'employee' is complex type.
<xs:sequence> : It defines that the complex type is a sequence of elements.
<xs:element name="firstname" type="xs:string"/> : It defines that the element 'firstname' is of string/text type.
<xs:element name="lastname" type="xs:string"/> : It defines that the element 'lastname' is of string/text type.
<xs:element name="email" type="xs:string"/> : It defines that the element 'email' is of

string/text type.

XML Schema Data types

There are two types of data types in XML schema.

1. simpleType
2. complexType

simpleType

The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.

complexType

The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

Result:

Ex.No: 11 **NOSQL DATABASE TOOLS**

Date:

Aim:

To Create Document, column and graph based data using NOSQL database tools.

Procedure:

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.
The following table shows the relationship of RDBMS terminology with MongoDB.

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	mongod
mysql/sqlplus	mongo

Sample Document

Following example shows the document structure of a blog site, which is simply a comma separated key value pair.

```
{
  _id: ObjectId(7df78ad8902c)
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user:'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
    },
    {
      user:'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
    }
  ]
}
```

_id is a 12 bytes hexadecimal number which assures the uniqueness of every document. You can provide _id while inserting the document. If you don't provide then MongoDB provides a unique id for every document. These 12 bytes first 4 bytes for the current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDB server and remaining 3 bytes are simple incremental VALUE.

The use Command

MongoDB **use DATABASE_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax

Basic syntax of **use DATABASE** statement is as follows −

use DATABASE_NAME

Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

```
>use mydb
switched to db mydb
```

To check your currently selected database, use the command **db**

```
>db
mydb
```

If you want to check your databases list, use the command **show dbs**.

```
>show dbs
local     0.78125GB
test      0.23012GB
```

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

```
>db.movie.insert({"name":"tutorials point"})
>show dbs
local      0.78125GB
mydb       0.23012GB
test       0.23012GB
```

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

The dropDatabase() Method

MongoDB **db.dropDatabase()** command is used to drop a existing database.

Syntax

Basic syntax of **dropDatabase()** command is as follows −

```
db.dropDatabase()
```

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example

First, check the list of available databases by using the command, **show dbs**.

```
>show dbs
local      0.78125GB
mydb       0.23012GB
test       0.23012GB
>
```

If you want to delete new database **<mydb>**, then **dropDatabase()** command would be as follows −

```
>use mydb
switched to db mydb
>db.dropDatabase()
>{ "dropped" : "mydb", "ok" : 1 }
>
```

Now check list of databases.

```
>show dbs
local      0.78125GB
test       0.23012GB
>
```

The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

Syntax

The basic syntax of **insert()** command is as follows −

```
>db.COLLECTION_NAME.insert(document)
```

Example

```
> db.users.insert({
... _id : ObjectId("507f191e810c19729de860ea"),
... title: "MongoDB Overview",
... description: "MongoDB is no sql database",
... by: "tutorials point",
```

```
... url: "http://www.tutorialspoint.com",
... tags: ['mongodb', 'database', 'NoSQL'],
... likes: 100
... })
WriteResult({ "nInserted" : 1 })
>
```

Here **mycol** is our collection name, as created in the previous chapter. If the collection doesn't exist in the database, then MongoDB will create this collection and then insert a document into it.

In the inserted document, if we don't specify the _id parameter, then MongoDB assigns a unique ObjectId for this document.

_id is 12 bytes hexadecimal number unique for every document in a collection. 12 bytes are divided as follows −

_id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)

You can also pass an array of documents into the insert() method as shown below:.

```
> db.createCollection("post")
> db.post.insert([
        {
                title: "MongoDB Overview",
                description: "MongoDB is no SQL database",
                by: "tutorials point",
                url:  "http://www.tutorialspoint.com",
                tags: ["mongodb", "database", "NoSQL"],
                likes: 100
        },
        {
        title: "NoSQL Database",
        description: "NoSQL database doesn't have tables",
        by: "tutorials point",
        url:  "http://www.tutorialspoint.com",
        tags: ["mongodb", "database", "NoSQL"],
        likes: 20,
        comments: [
                {
                        user:"user1",
                        message: "My first comment",
                        dateCreated: new Date(2013,11,10,2,35),
                        like: 0
                }
        ]
}
])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
```

```
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

To insert the document you can use **db.post.save(document)** also. If you don't specify **_id** in the document then **save()** method will work same as **insert()** method. If you specify _id then it will replace whole data of document containing _id as specified in save() method.

The insertOne() method

If you need to insert only one document into a collection you can use this method.

Syntax

The basic syntax of insert() command is as follows −

```
>db.COLLECTION_NAME.insertOne(document)
```

Example

Following example creates a new collection named empDetails and inserts a document using the insertOne() method.

```
> db.createCollection("empDetails")
{ "ok" : 1 }
> db.empDetails.insertOne(
        {
                First_Name: "Radhika",
                Last_Name: "Sharma",
                Date_Of_Birth: "1995-09-26",
                e_mail: "radhika_sharma.123@gmail.com",
                phone: "9848022338"
        })
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5dd62b4070fb13eec3963bea")
}
>
```

The insertMany() method

You can insert multiple documents using the insertMany() method. To this method you need to pass an array of documents.

Example

Following example inserts three different documents into the empDetails collection using the insertMany() method.

```
> db.empDetails.insertMany(
        [
                {
                        First_Name: "Radhika",
                        Last_Name: "Sharma",
                        Date_Of_Birth: "1995-09-26",
                        e_mail: "radhika_sharma.123@gmail.com",
                        phone: "9000012345"
                },
                {
                        First_Name: "Rachel",
                        Last_Name: "Christopher",
```

```
                        Date_Of_Birth: "1990-02-16",
                        e_mail: "Rachel_Christopher.123@gmail.com",
                        phone: "9000054321"
                },
                {
                        First_Name: "Fathima",
                        Last_Name: "Sheik",
                        Date_Of_Birth: "1990-02-16",
                        e_mail: "Fathima_Sheik.123@gmail.com",
                        phone: "9000054321"
                }
        ]
)
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5dd631f270fb13eec3963bed"),
                ObjectId("5dd631f270fb13eec3963bee"),
                ObjectId("5dd631f270fb13eec3963bef")
        ]
}
>
```

Result:

Thus the Document, column and graph based data using NOSQL database tools is created successfully.

Ex.No: 12 **EMPLOYEE MANAGEMENT SYSTEM**

Date:

Aim:

To create an Employee Management System

Algorithm:

1. Create a database table named "Employee" with columns empno, empname, department, and salary using the SQL query: create table Employee(empno INT primary key, empname varchar(20), department varchar(20), salary INT)
2. Establish a database connection using the JDBC API with the following details:
    - Database: Oracle
    - Driver: ojdbc7
    - Service ID: orcl
    - Username: system
    - Password: manager
3. Create a Java GUI using NetBeans IDE with the following components:
    - Labels and text fields for empno, empname, department, and salary
    - Buttons for adding, searching, updating, and deleting employee records
    - Option pane for displaying messages
4. Write Java code to handle the actions of the GUI components:
5. Display appropriate messages using option pane for success, failure or error during the execution of database operations.

Close the database connection and handle any errors that may occur

Tables :

Sql> create table Employee(empno INT primary key , empname varchar(20) ,department varchar(20) , salary INT)

Sql>select *from employee;

EMPID	EMPNAME	EMPDEPT	SALARY
1	rekha	cse	30000
2	renu	cse	70000
3	elamathi	ece	90000
4	rahul	ece	79990

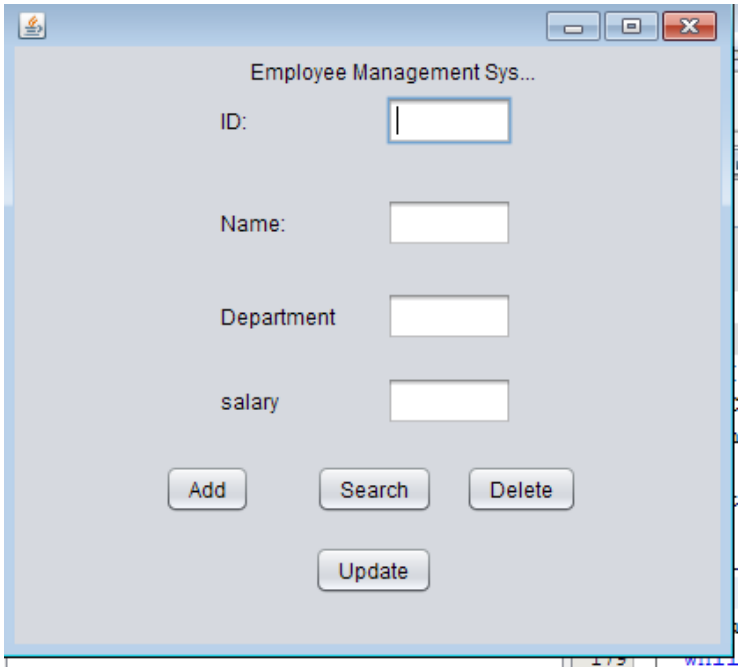
Net beans:

Add jar file.

Right click on package---select properties-----select libraries- add jar file(ojdbc7).

Right click on package --- select new ---select JFrame forms.

Design forms:



Procedure:

1. Code for components in GUI will come automatically.
2. Do oracle connectivity. (click on services ---right click on database—select new connection—
   Select oracle thin driver----give service id as orcl.---give username and password.---click test connection.

3. Import javax.swing and java.sql
4. Double click on button in forms and type the coding

Coding:

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
try{
  Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:orcl","system","manager");

Statement stmt=con.createStatement();

String a=JOptionPane.showInputDialog(null,"Enter Employee id");
int temp=0;
ResultSet rs=stmt.executeQuery("select * from employee");
while(rs.next())
{
   if (rs.getString(1).equals(a))
   {
      temp=1;


         }
}
```

```
if (temp==1)
{
   String query="delete from employee where empid= " + a ;
   stmt.execute(query);
      JOptionPane.showMessageDialog (null,"employee record deleted");

}
else

     JOptionPane.showMessageDialog (null,"employee record not found");



con.close();

}

catch(SQLException ex)
      {
       JOptionPane.showMessageDialog (null,ex);

      } catch (ClassNotFoundException ex) {

      }


   }

   private void salaryActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
   }

   private void nameActionPerformed(java.awt.event.ActionEvent evt) {
      // TODO add your handling code here:
   }

   private void addbuttonActionPerformed(java.awt.event.ActionEvent evt) {
try
{

    Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:orcl","system","manager");

Statement stmt=con.createStatement();
String query="insert into employee values( "
      + id.getText()+", "'+name.getText()+"' , "'+department.getText()+"' ,"
     +salary.getText()+")";

//String query ="insert into employee values(34,'mala','cse',234324)";
      stmt.execute(query);



      JOptionPane.showMessageDialog (null,"employee added");
```

```
        id.setText(null);
        name.setText(null);
        department.setText(null);
        salary.setText(null);

//con.close();

}

catch(SQLException ex)
        {
         JOptionPane.showMessageDialog (null,ex);

        } catch (ClassNotFoundException ex) {
        }
    }

    private void addbuttonMouseClicked(java.awt.event.MouseEvent evt) {

    }
```

```
   private void searchActionPerformed(java.awt.event.ActionEvent evt) {
try{

      Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:orcl","system","manager");

Statement stmt=con.createStatement();
   String a=JOptionPane.showInputDialog(null,"Enter Employee id");

ResultSet rs=stmt.executeQuery("select * from employee");
while(rs.next())
{
   if (rs.getString(1).equals(a))
   {
      id.setText(rs.getString(1));
            name.setText(rs.getString(2));
      department.setText(rs.getString(3));
      salary.setText(rs.getString(4));


   }
}


   }
   catch(SQLException ex)
      {
       JOptionPane.showMessageDialog (null,ex);

      } catch (ClassNotFoundException ex) {


      }
   }
   private void updateActionPerformed(java.awt.event.ActionEvent evt) {
try
{
    Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(
"jdbc:oracle:thin:@localhost:1521:orcl","system","manager");
Statement stmt=con.createStatement();
String query="update employee set empid= " + id.getText()+",empname='"
      +name.getText()+"' , empdept='"+department.getText()+"' ,salary="
     +salary.getText()+"where empid= " + id.getText();

//String query ="insert into employee values(34,'mala','cse',234324)";
      stmt.executeQuery(query);
      JOptionPane.showMessageDialog (null,"employee details updated");
      id.setText(null);
      name.setText(null);
      department.setText(null);
      salary.setText(null);
      con.close();
```

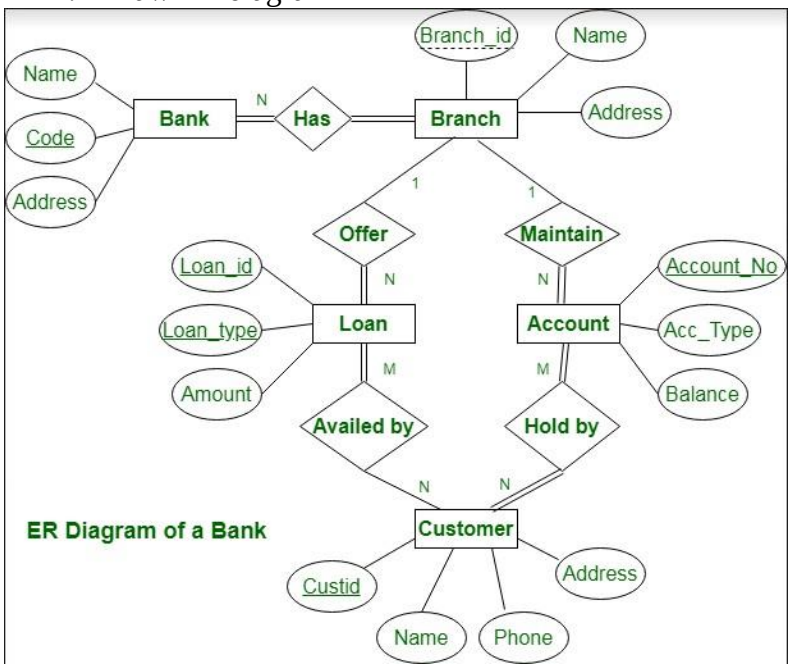
```
        }
catch(SQLException ex)
        {
          JOptionPane.showMessageDialog (null,ex);

        } catch (ClassNotFoundException ex) {
      }
    }
```

Result:

Ex.No: 13 **CASE STUDY**

Date: **BANK MANAGEMENT SYSTEM**

1. Draw ER diagram



2. Create database tables
3. Design forms
4. Complete the connectivity.
5. Run the project

Result:

MADHA ENGINEERING COLLEGE

(Affiliated to Anna University and Approved by AICTE, New Delhi)
Madha Nagar, Kundrathur, Chennai-600069

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMMON TO: DEPARTMENT OF INFORMATION TECHNOLOGY

CS3461– OPERATING SYSTEMS LABORATORY

R2021

LAB MANUAL

INDEX

S.No.	Name of the Experiment	Page No	Date of Experiment	Remarks
1.	Installation Of Windows Operating System			
2.	A. Illustrate Unix Commands And Shell Programming B. Shell Programming			
3.	A. Process Management Using System Calls : Fork, Exit, Getpid, Wait, Close B. Program Using System Calls Wait( ) & Exit( )			
4.	Write C programs to implement the various CPU Scheduling Algorithms			
5.	Illustrate The Inter Process Communication Strategy			
6.	Implement Mutual Exclusion By Semaphore			
7.	Write C Programs To Avoid Deadlock Using Banker's Algorithm			
8.	Write A C Program To Implement Deadlock Detection Algorithm			
9.	Write C Program To Implement Threading			

10.	Implement The Paging Technique Using C Program			
11.	Write C Programs To Implement The Following Memory Allocation Methods			
12.	Write C Programs To Implement The Various Page Replacement Algorithms First In First Out (FIFO) Page Replacement Algorithm:			
13.	Write C Programs To Implement The Various File Organization Techniques			
14	Implemention File Allocation Strategies Using C Programs			
15.	Write C Programs For The Implementation Of Various Disk Scheduling Algorithm			
16.	Install Any Guest Operating System Like Linux Using Vmware			

Exp.No:1 **INSTALLATION OF WINDOWS OPERATING SYSTEM**

AIM:

To Study the Installation of Windows Operating System.

PROCEDURE:

Steps to install Windows 10 :

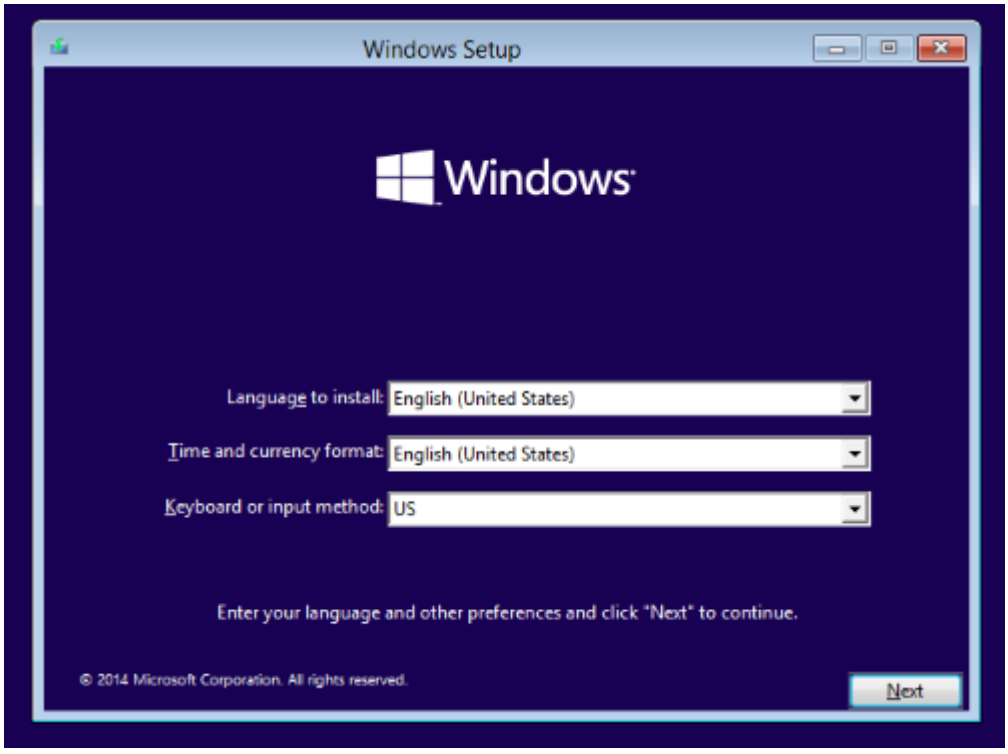
Step 1: Download Windows 10 ISO image from http://windows.microsoft.com/en-us/windows/preview-iso. Read more about Windows 10 technical preview ISO images.

Step 2: Mount the ISO image as a drive or burn it into a DVD. Optionally, you can create a Windows 10 bootable USB drive.

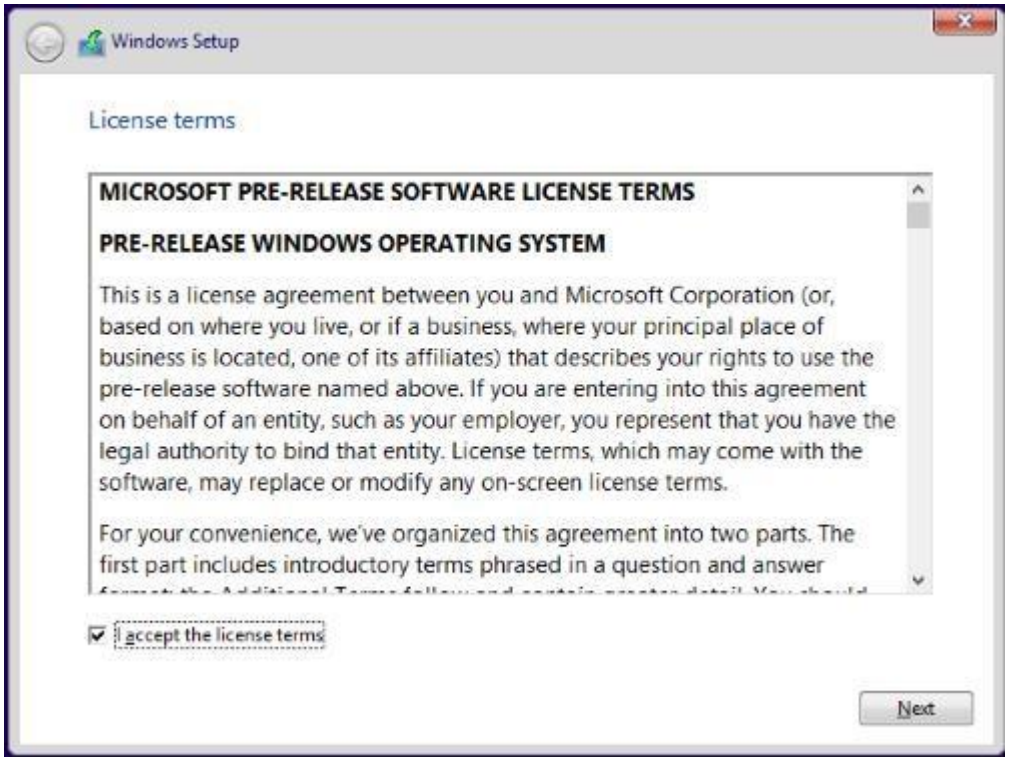
Step 3: Launch your virtual machine software and create a new virtual machine. You can read more details on how to install Windows 10 on VMWare Workstation.

Step 4: After you create the virtual machine, follow the steps below to install Windows 10.

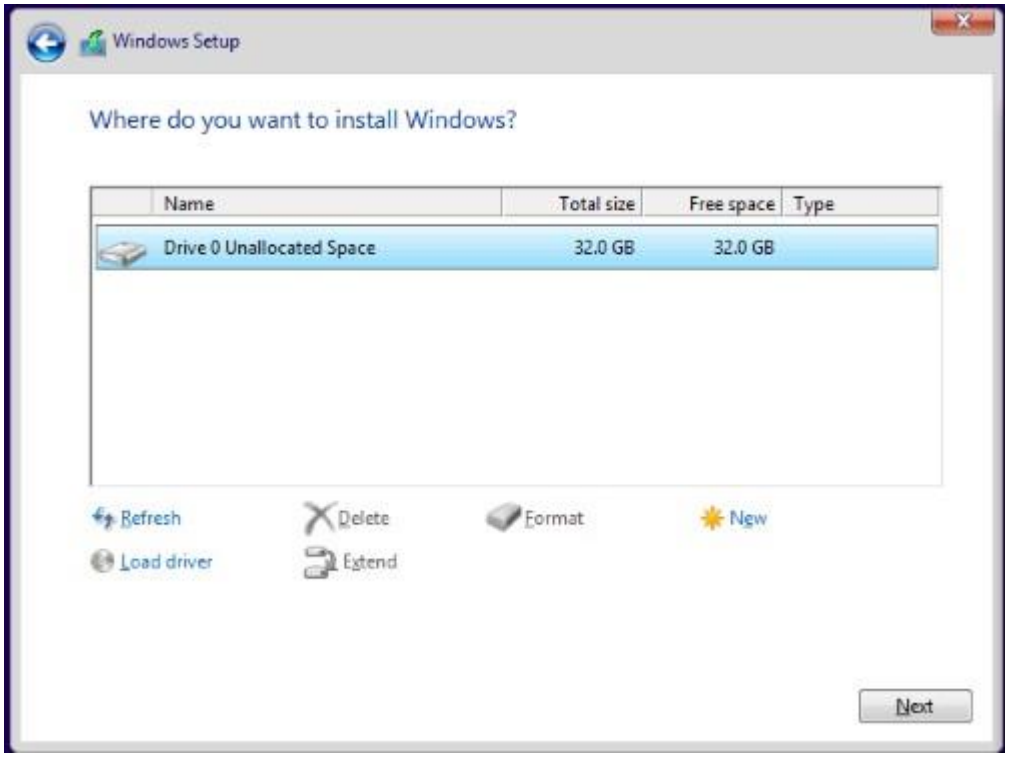
Step 5: In the welcome screen, choose your language settings and click the "Next" button to proceed.



Step 6: Accept the license terms and click "Next" to proceed.

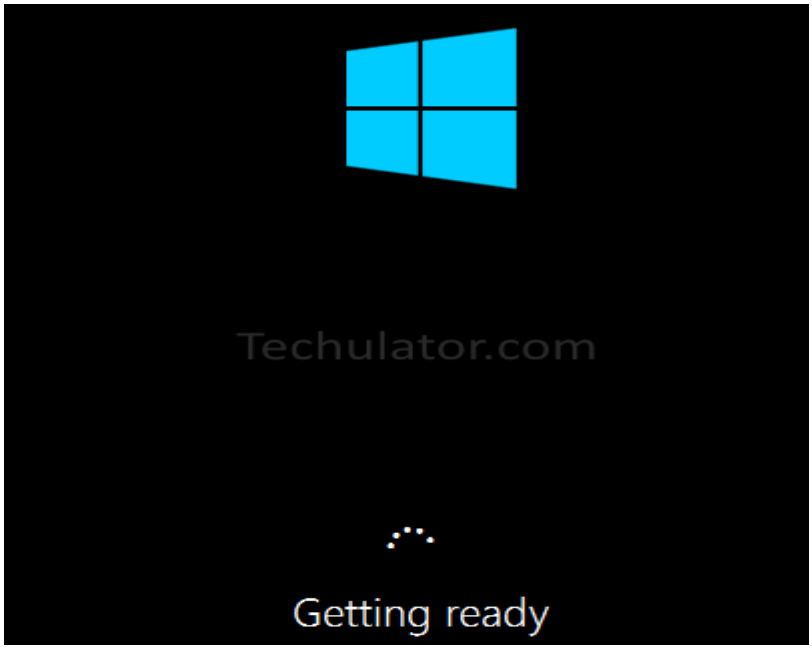


Step 7: In the next step, you have to select the drive to install. If you have multiple drives on your computer, you will see them all there. Choose the correct drive to install. If you are installing Windows 10 over an existing Windows, you may install side by side on another drive.



Step 8: Installer will copy all the necessary files to the computer and continue with the installation. Depending on your system configuration, it may take a while (10-20 minutes) to complete the installation.

After successful installation, you will see the Windows welcome screen, which is similar to the Windows 8 theme.



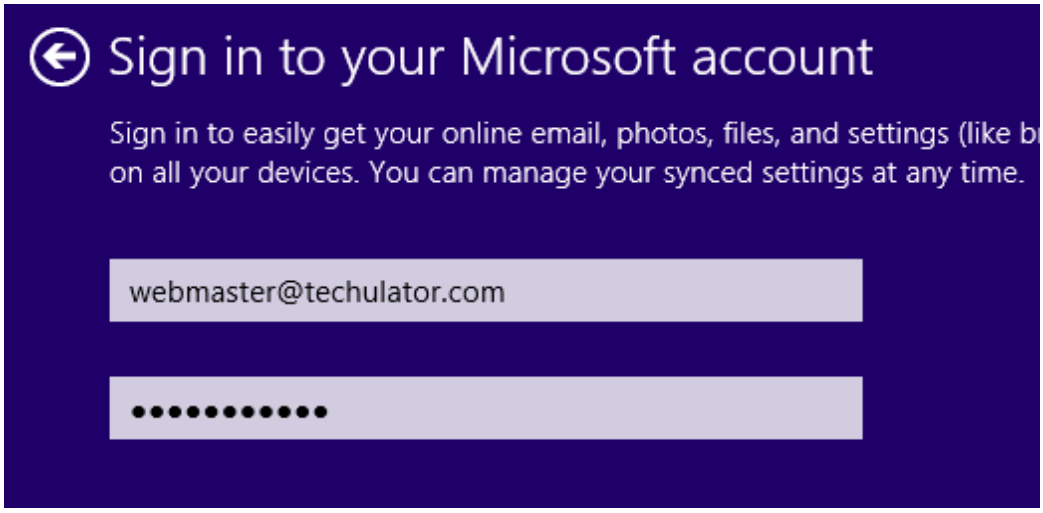
You will be presented with an option to select the Settings. Unless you want to make custom

configuration, you can choose "Express Settings" and proceed.

In the next step, you will be prompted to select One Drive option. If you don't want to integrate One Drive storage for this computer, you can Turn Off the One Drive options.

The next step is installing some basic apps for your Windows 10 computer. No intervention from your side is necessary. Windows will automatically install the apps for you. It may take several minutes to complete the installation.

As part of installation, you will need to connect your Microsoft account with your OS to access various services like Microsoft App Store. You will need to use the same Microsoft account to login to your Windows.



You are all set. In a few minutes, Windows 10 will be configured and will be ready for you to use. If you encounter any errors or have any suggestions for Microsoft to improve the OS, don't forget to report them to Microsoft.

RESULT:

Exp.No:2a **ILLUSTRATE UNIX COMMANDS AND SHELL PROGRAMMING**

AIM:

To Illustrate UNIX commands

ALGORITHM:

Here's a general algorithm for executing a UNIX command step by step:

1. Open a UNIX terminal window.
2. Type the name of the command you want to execute, followed by any necessary arguments.
3. Press Enter to execute the command.
4. The command will run, and any output will be displayed in the terminal window.
5. Once the command has completed, the terminal will return to the command prompt.

Here are some examples of UNIX commands and their usage:

1. ls - List the contents of the current directory
    1. Open a UNIX terminal window.
    2. Type "ls" and press Enter.
    3. The contents of the current directory will be displayed in the terminal window.
2. cd - Change the current directory
    1. Open a UNIX terminal window.
    2. Type "cd [directory name]" and press Enter.
    3. The current directory will change to the specified directory.
3. mkdir - Create a new directory
    1. Open a UNIX terminal window.
    2. Type "mkdir [directory name]" and press Enter.
    3. A new directory with the specified name will be created in the current directory.
4. touch - Create a new file
    1. Open a UNIX terminal window.
    2. Type "touch [file name]" and press Enter.
    3. A new file with the specified name will be created in the current directory.

5. rm - Remove a file or directory
    1. Open a UNIX terminal window.
    2. Type "rm [file or directory name]" and press Enter.
    3. The specified file or directory will be removed from the system.

PROGRAM:

OUTPUT:

RESULT:

Exp.No:2b **SHELL PROGRAMMING**

AIM:

To write shell programs in C

ALGORITHM:

1. Include necessary header files: To write a shell program in C, you need to include the necessary header files for input/output, string manipulation, process management, and signal handling. The most commonly used header files are <stdio.h>, <stdlib.h>, <string.h>, <unistd.h>, and <signal.h>. Include these header files at the beginning of your program.
2. Read user input: The user input is read from the standard input stream using the fgets() function. You need to provide a buffer to store the user input and specify the maximum number of characters to be read. Parse the input string to extract the command and any arguments.
3. Fork a child process: To execute the command entered by the user, you need to create a child process using the fork() function. The child process will execute the command, while the parent process waits for the child process to finish.
4. Execute the command: In the child process, you need to use the execvp() function to execute the command entered by the user. This function takes two arguments: the command to be executed and an array of strings containing the command-line arguments.
5. Handle signals: Signals are used to interrupt or terminate a process. In your shell program, you need to handle signals using the signal() function. For example, you can handle the Ctrl+C signal (SIGINT) by sending a signal to the child process to terminate it.
6. Wait for the child process: In the parent process, you need to wait for the child process to finish using the waitpid() function. This function waits for a specific child process to finish and returns the status of the child process.
7. Repeat the process: After the child process has finished executing the command, you need to repeat the process of reading user input, forking a child process, executing the command, and waiting for the child process to finish. This process continues until the user enters the exit command or the program is terminated.
8. Clean up resources: When the program exits, you need to clean up any resources that were allocated during its execution, such as file descriptors, memory, and child processes.

PROGRAM:

OUTPUT:

RESULT:

Exp.No: 3 PROCESS MANAGEMENT USING SYSTEM CALLS : FORK, EXIT, GETPID, WAIT, CLOSE

AIM :

To write the program to create a Child Process using system call fork().

ALGORITHM :

Step 1 : Declare the variable pid.

Step 2 : Get the pid value using system call fork().

Step 3 : If pid value is less than zero then print as “Fork failed”.

Step 4 : Else if pid value is equal to zero include the new process in the system‟s file using execlp system call.

Step 5 : Else if pid is greater than zero then it is the parent process and it waits till the child completes using the system call wait()

Step 6 : Then print “Child complete”.

SYSTEM CALLS USED:

1. fork( )

Used to create new processes. The new process consistsof a copy of the address space of the original process. The value of process id for the child process is zero, whereas the value of process id for the parent is an integer value greater than zero.

Syntax : fork( )

2.execlp( )

Used after the fork() system call by one of the two processes to replace the process‟ memory space with a new program. It loads a binary file into memory destroying the memory image of the program containing the execlp system call and starts its execution.The child process overlays its address space with the UNIX command /bin/ls using the execlp system call.

Syntax : execlp( )

3. wait( )

The parent waits for the child process to complete using the wait system call. The wait system call returns the process identifier of a terminated child, so that the parent can tell which of its possibly many children has terminated.

Syntax : wait(NULL)

4. exit( )

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit system call. At that point, the process may return data (output) to its parent process (via the wait system call).

Syntax: exit(0)

PROGRAM:

OUTPUT:

RESULT:

Exp.NO:3b PROGRAM USING SYSTEM CALLS wait( ) & exit( )

AIM :

To write the program to implement the system calls wait( ) and exit( ).

ALGORITHM :

Step 1 : Declare the variables pid and i as integers.

Step 2 : Get the child id value using the system call fork().

Step 3 : If child id value is less than zero then print “fork failed”.

Step 4 : Else if child id value is equal to zero , it is the id value of the child and then start the child process to execute and perform Steps 6 & 7.

Step 5 : Else perform Step 8.

Step 6 : Use a for loop for almost five child processes to be called.

Step 7 : After execution of the for loop then print “child process ends”.

Step 8 : Execute the system call wait( ) to make the parent to wait for the child process to get over.

Step 9 : Once the child processes are terminated , the parent terminates and hence print “Parent process ends”.

Step 10 : After both the parent and the chid processes get terminated it execute the wait( ) system call to permanently get deleted from the OS.

SYSTEM CALL USED :

1. fork ( )

   Used to create new process. The new process consists of a copy of the address space of the original process. The value of process id for the child process is zero, whereas the value of process id for the parent is an integer value greater than zero.

   Syntax: fork ( )

2. wait ( )

   The parent waits for the child process to complete using the wait system call. The wait system call returns the process identifier of a terminated child, so that the parent can tell which of its possibly many children has terminated.

Syntax: wait (NULL)

3. exit ( )

A process terminates when it finishes executing its final statement and asks the operating system to delete it by using the exit system call. At that point, the process may return data (output) to its parent process (via the wait system call).

Syntax: exit(0)

PROGRAM:

OUTPUT:

RESULT:

Exp.No:4 Write C programs to implement the various CPU Scheduling Algorithms

A.FIRST COME FIRST SERVE:

AIM:

To write a c program to simulate the CPU scheduling algorithm First Come First Serve (FCFS

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process name and the burst Time

Step 4: Set the waiting of the first process as _0'and its burst time as its turnaround time

Step 5: for each process in the Ready Q calculate

Waiting time (n) = waiting time (n-1) + Burst time (nTurnaround time (n) = waiting time (n)+Burst time(n)

Step 6: Calculate

Average waiting time = Total waiting Time / Number of process

Average Turnaround time = Total Turnaround Time / Number of process

Step 7:stop

PROGRAM:

OUTPUT:

RESULT:

B.SHORTEST JOB FIRST:

AIM:

To write a program to stimulate the CPU scheduling algorithm Shortest job first (Non- Preemption)

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as _0' and its turnaround time as its burst time.

Step 6: Sort the processes names based on their Burt time

Step 7: For each process in the ready queue,

calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 8: Calculate

c)Average waiting time = Total waiting Time / Number of process

d)Average Turnaround time = Total Turnaround Time / Number of process Step

9: Stop the process

PROGRAM:

OUTPUT:

RESULT:

C.ROUND ROBIN:

AIM:

To simulate the CPU scheduling algorithm round-robin.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process (n) = burst time process (n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q, calculate

a) Waiting time for process (n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU fromprocess(n-1)

b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7: Calculate

c) Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number ofprocess

Step 8: Stop the process

PROGRAM:

OUTPUT:

RESULT:

D.PRIORITY:

AIM:

To write a c program to simulate the CPU scheduling priorityalgorithm.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Sort the ready queue according to the priority number.

Step 5: Set the waiting of the first process as _0' and its burst time as its turnaround time

Step 6: Arrange the processes based on process priority

Step 7: For each process in the Ready Q calculate Step 8:

for each process in the Ready Q calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 9: Calculate

Average waiting time = Total waiting Time / Number of process

c) Average Turnaround time = Total Turnaround Time / Number of process Print the results in an order.

Step10: Stop

PROGRAM:

OUTPUT:

RESULT:

STRATEGY

AIM:

The aim of this program is to create two child processes and establish communication between them using pipes. One child process will write a message to the pipe, and the other child process will read the message from the pipe.

ALGORITHM:

1. Create a pipe using the **pipe()** system call.
2. Create two child processes using the **fork()** system call.
3. In the parent process, close both ends of the pipe and wait for both child processes to exit.
4. In the first child process, close the reading end of the pipe and write a message to the writing end of the pipe.
5. In the second child process, close the writing end of the pipe and read the message from the reading end of the pipe.
6. Print the message in the second child process.
7. Exit both child processes.

PROGRAM:

OUTPUT:

RESULT:

Exp.No:6 IMPLEMENT MUTUAL EXCLUSION BY SEMAPHORE

Aim:

The aim of this program is to demonstrate mutual exclusion using semaphores. Two threads will try to access a shared resource, and the semaphore will ensure that only one thread accesses the resource at a time.

Algorithm:

1. Define a shared resource that needs to be accessed mutually exclusively.
2. Define a semaphore variable and initialize it with the value 1.
3. Create two threads that will access the shared resource.
4. In each thread, wait for the semaphore to become available by calling sem_wait().
5. Once the semaphore is available, access the shared resource and perform the necessary operations.
6. Release the semaphore by calling sem_post().
7. Join the two threads and exit.

PROGRAM:

OUTPUT:

RESULT:

Exp.No: 7 WRITE C PROGRAMS TO AVOID DEADLOCK USING BANKER'S ALGORITHM

Aim:

To avoid deadlock using Banker's Algorithm in a C program.

Algorithm:

1. Take input for the number of processes and the number of resources.
2. Take input for the maximum demand of each process and the current allocation of each process.
3. Calculate the need matrix for each process.
4. Initialize the work array with the available resources.
5. Initialize the finish array to false for all processes.
6. Repeat steps 7 to 9 until all processes are finished.
7. Find a process that is not finished and whose need can be satisfied with the available resources.
8. If such a process is found, allocate the resources to the process and mark it as finished.
9. If no such process is found, the system is in deadlock.

PROGRAM

OUTPUT

RESULT

Exp.No:8 WRITE A C PROGRAM TO IMPLEMENT DEADLOCK DETECTION ALGORITHM

Aim:

The aim of this program is to demonstrate mutual exclusion using semaphores. Two threads will try to access a shared resource, and the semaphore will ensure that only one thread accesses the resource at a time.

Algorithm:

1. Define a shared resource that needs to be accessed mutually exclusively.
2. Define a semaphore variable and initialize it with the value 1.
3. Create two threads that will access the shared resource.
4. In each thread, wait for the semaphore to become available by calling **sem_wait()**.
5. Once the semaphore is available, access the shared resource and perform the necessary operations.
6. Release the semaphore by calling **sem_post()**.
7. Join the two threads and exit.

Program:

Output:

Result:

Exp.No:9 WRITE C PROGRAM TO IMPLEMENT THREADING

Aim:

To implement threading in a C program.

Algorithm:

1. Create a function that will be executed by the thread.
2. Declare a thread identifier variable.
3. Call the thread creation function with the thread identifier variable, the thread attributes, the function to be executed, and any arguments to be passed to the function.
4. Wait for the thread to complete using the join function.
5. Exit the program.

Program:

Output:

Result:

Exp.No:10 IMPLEMENT THE PAGING TECHNIQUE USING C PROGRAM

Aim:

To implement the paging technique in a C program.

Algorithm:

1. Take input for the size of the main memory, the size of each page, and the size of the process.
2. Calculate the number of pages required for the process.
3. Initialize the page table with the page numbers and the corresponding frame numbers as -1.
4. Repeat steps 5 to 9 for each page of the process.
5. Calculate the page number for the current page of the process.
6. Check if the page is already in the main memory by searching the page table for the page number.
7. If the page is not in the main memory, find a free frame in the main memory.
8. If a free frame is found, update the page table with the page number and the corresponding frame number.
9. If no free frame is found, use a page replacement algorithm to replace a page in the main memory with the current page of the process.
10. Repeat steps 4 to 9 for each page of the process.

Program:

Output:

Result:

Ex No:11 Write C programs to implement the following Memory Allocation Methods

Aim:

To write C programs to implement the following Memory Allocation Methods

a. First Fit b. Worst Fit c. Best Fit

Algorithm:

a. First-Fit Allocation Algorithm:

Step 1: Traverse through the free list from beginning to end.

Step 2: If a block of memory is found that is big enough to accommodate the requested memory, allocate it and return a pointer to the beginning of the block.

Step 3: If no such block is found, return an error indicating that the request cannot be satisfied.

b. Best-Fit Allocation Algorithm:

Step 1: Traverse through the free list to find the smallest block that can fit the requested memory size

Step 2: Allocate that block and return a pointer to the beginning of the block

Step 3: If no such block is found, return an error indicating that the request cannot be satisfied

c. Worst-Fit Allocation Algorithm:

Step 1: Traverse through the free list to find the largest block that can fit the requested memory size

Step 2: Allocate that block and return a pointer to the beginning of the block

Step 3: If no such block is found, return an error indicating that the request cannot be satisfied

Program:

Output:

Result:

Ex No:12 **Write C programs to implement the various Page Replacement Algorithms**

First In First Out (FIFO) Page Replacement Algorithm:

Aim:

To implement the FIFO page replacement algorithm in a C program.

Algorithm:

1. Take input for the size of the main memory, the size of each page, and the size of the process.
2. Calculate the number of pages required for the process.
3. Initialize the page table with the page numbers and the corresponding frame numbers as -1.
4. Initialize the frame queue to an empty state.
5. Repeat steps 6 to 10 for each page of the process.
6. Calculate the page number for the current page of the process.
7. Check if the page is already in the main memory by searching the page table for the page number.
8. If the page is not in the main memory, find a free frame in the main memory.
9. If a free frame is found, update the page table with the page number and the corresponding frame number.
10. If no free frame is found, replace the page at the front of the frame queue with the current page of the process and update the page table.
11. Add the current page of the process to the back of the frame queue.
12. Repeat steps 5 to 11 for each page of the process.

Program:

Output:

Result:

Ex No:13 Write C programs to Implement the various File Organization Techniques

Aim:

To write C programs to Implement the various File Organization Techniques.

Algorithm:

1)Sequential File Organization

Step 1: Open the file for writing

Step 2: Write the record to the file

Step 3: Close the file

Step 4: Open the file for reading

Step 5: Read the record from the file

Step 6: Close the file

b) Indexed File Organization

Step 1: Open the file for writing

Step 2: Write the record to the file

Step 3: Close the file

Step 4: Open the index file for writing

Step 5: Write the index record to the index file

Step 6: Close the index file

Step 7: Open the index file for reading

Step 8: Search the index file for the desired record

Step 9: Retrieve the record from the data file using the index

Step 10: Close the index file

Program:

Output:

Result:

Ex No:14 Implemention File Allocation Strategies using C programs

Aim:

. Implement the following File Allocation Strategies using C programs

a. Sequential b. Indexed c. Linked

Algorithm:

a. Sequential

Step 1: Start the program.
Step 2: Get the number of memory partition and their sizes.
Step 3: Get the number of processes and values of block size for each process.
Step 4: First fit algorithm searches all the entire memory block until a hole which is big enough is encountered. It allocates that memory block for the requesting process.
Step 5: Best-fit algorithm searches the memory blocks for the smallest hole which can be allocated to requesting process and allocates it.
Step 6: Worst fit algorithm searches the memory blocks for the largest hole and allocates it to the process.
Step 7: Analyses all the three memory management techniques and display the best algorithm which utilizes the memory resources effectively and efficiently.
Step 8: Stop the program.

b. Indexed:

Step 1: Start.
Step 2: Let n be the size of the buffer
Step 3: check if there are any producer
Step 4: if yes check whether the buffer is full
Step 5: If no the producer item is stored in the buffer
Step 6: If the buffer is full the producer has to wait
Step 7: Check there is any consumer.If yes check whether the buffer is empty
Step 8: If no the consumer consumes them from the buffer
Step 9: If the buffer is empty, the consumer has to wait.
Step 10: Repeat checking for the producer and consumer till required
Step 11: Terminate the process.

c. Linked:

Step 1: Create a queue to hold all pages in memory
Step 2: When the page is required replace the page at the head of the queue
Step 3: Now the new page is inserted at the tail of the queue
Step 4: Create a stack

Step 5: When the page fault occurs replace page present at the bottom of the stack

Step 6: Stop the allocation.

Program:

Output:

Result:

Ex No:15 Write C programs for the implementation of various disk scheduling

Algorithm

Aim:

To write C programs for the implementation of various disk scheduling algorithm.

Algorithm:

a .FCFS (First-Come, First-Serve) Disk Scheduling Algorithm:

Step 1.Read the size of the queue and the queue elements from the user.

Step 2.Read the head position from the user.

Step 3.Process the requests in the order in which they are received.

Step 4.Calculate the seek time as the distance traveled by the head from one request to another.

b. SSTF (Shortest Seek Time First) Disk Scheduling Algorithm:

Step 1.Start by reading the size of the disk queue, the disk queue itself, and the initial head position from the user.

Step 2.Initialize a variable "total" to 0 to keep track of the total head movement.

Step 3.Loop through the disk queue:

1. Find the position of the disk request with the shortest seek time (i.e., the smallest absolute difference between the head position and the request position).
2. Swap the disk request at the current index with the disk request at the found position.
3. Add the absolute difference between the head position and the new request position to "total".
4. Update the head position to the new request position.

Step 4.Print out the total head movement.

c . SCAN Disk Scheduling Algorithm:

Step 1.Read the size of the disk queue, the disk queue itself, the initial head position, and the direction of movement from the user.

Step 2.Add the head position to the end of the disk queue.

Step 3.Sort the disk queue in ascending or descending order depending on the direction of movement.

Step 4.Find the position of the head in the sorted disk queue.

Step 5.Print out the order in which the disk requests are serviced:

1. If the direction of movement is left, print out the requests from the head position to the beginning of the disk.

2. Then print out the requests from the head position to the end of the disk.

3. If the direction of movement is right, print out the requests from the head position to the end of the disk.

4. Then print out the requests from the head position to the beginning of the disk.

Program:

Output:

Result:

Ex no:16 Install any guest operating system like Linux using VMware

Aim:

To install any guest operating system like Linux using VMware

Algorithm:

Step:1 Download and install VMware on your computer.

Step:2 Open VMware and click on "Create a New Virtual Machine".

Step:3 Select "Typical" for the setup type and click "Next".

Step:4 Choose the installer disc image file (ISO) for the Linux operating system you want to install and click "Next".

Step:5 Enter a name for your virtual machine and choose a location to save it. Click "Next".

Step:6 Choose the maximum disk size for your virtual machine and select whether to store the disk as a single file or split it into multiple files. Click "Next".

Step:7 Customize the hardware settings for your virtual machine, such as the amount of RAM and number of processors. Click "Next".

Step:8 Review the summary of your virtual machine settings and click "Finish" to create the virtual machine.

Step:9 Start the virtual machine and follow the Linux installation wizard to install the operating system.

Program:

Choose Your Preferred Linux OS

You probably know which Linux OS you want to try. Some Linux distros are particularly suited to running in a VM, but others are not. All 32-bit and 64-bit distros work in a virtual machine. However, you cannot run Linux distros for ARM architecture (such as the Raspberry Pi) in VMware.

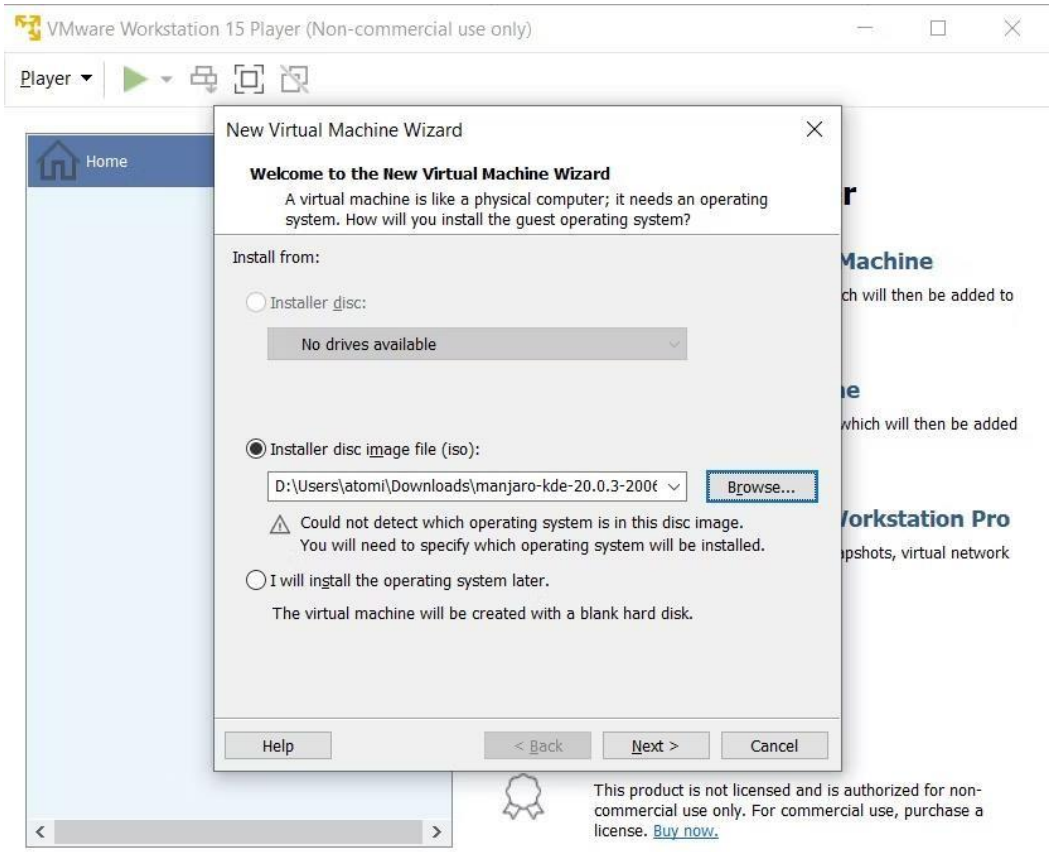
Should you want to emulate an ARM environment in Windows, try QEMU.

If you don't know which OS to choose, however, you'll find our regularly-updated list of the best Linux distributions here.

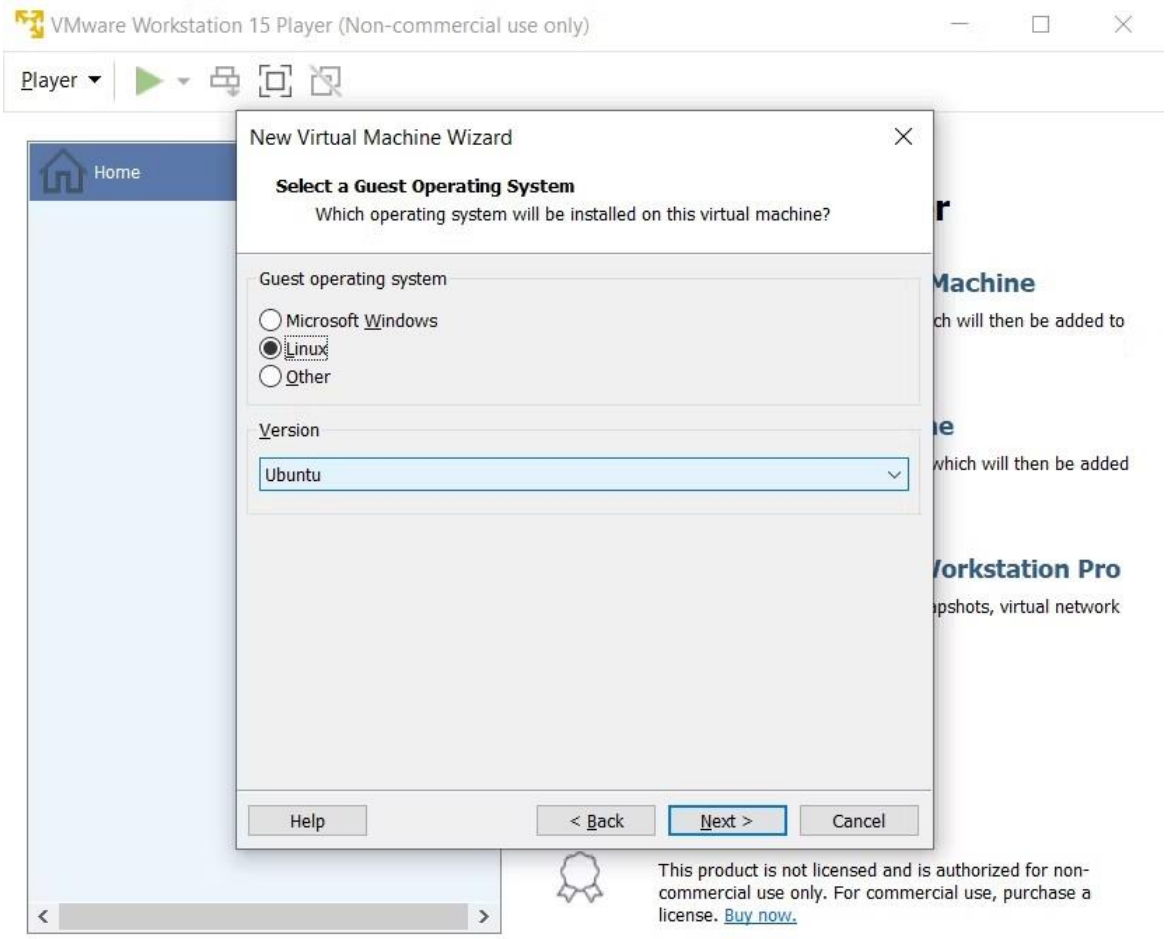
Create Your Linux Virtual Machine

While your Linux ISO downloads, it's a good time to start configuring your VM. Start by launching VMware Workstation Player. When you're ready to create a VM:

1. **Click Create a New Virtual Machine**
2. **Select the default option, Installer disc image file (iso)**
3. **Click Browse to find the ISO file**



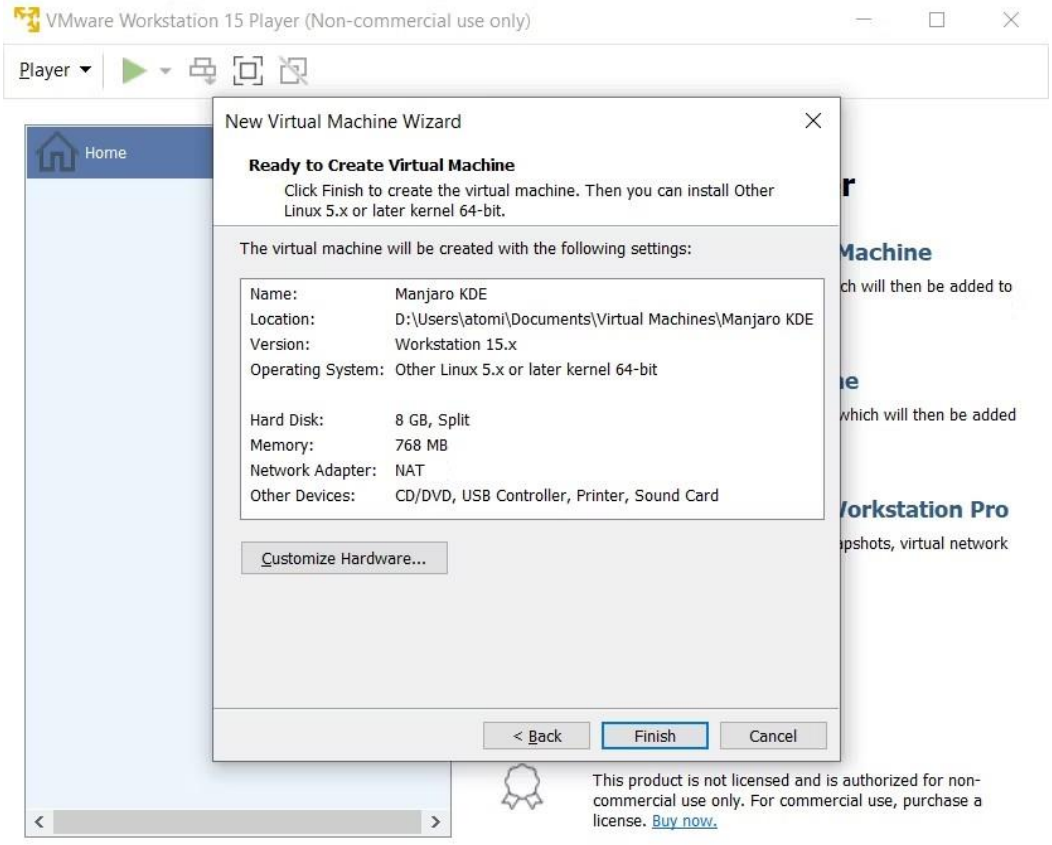
4. With "guest" OS selected, click **Next**
5. Select **Linux** as the Guest operating system type



6. Under **Version**, scroll through the list and select the OS
7. Click **Next** to proceed and if necessary, input a **Virtual machine name**
8. Confirm the storage **Location** and change if needed

With the operating system selected and configured, it's time to build the virtual machine.

9. Under **Specify Disk Capacity** adjust **Maximum disk size** if required (the default should be enough)
10. Select **Split virtual disk into multiple files** as this makes moving the VM to a new PC easy
11. Click **Next** then confirm the details on the next screen
12. If anything seems wrong click **Back**, otherwise click **Finish**

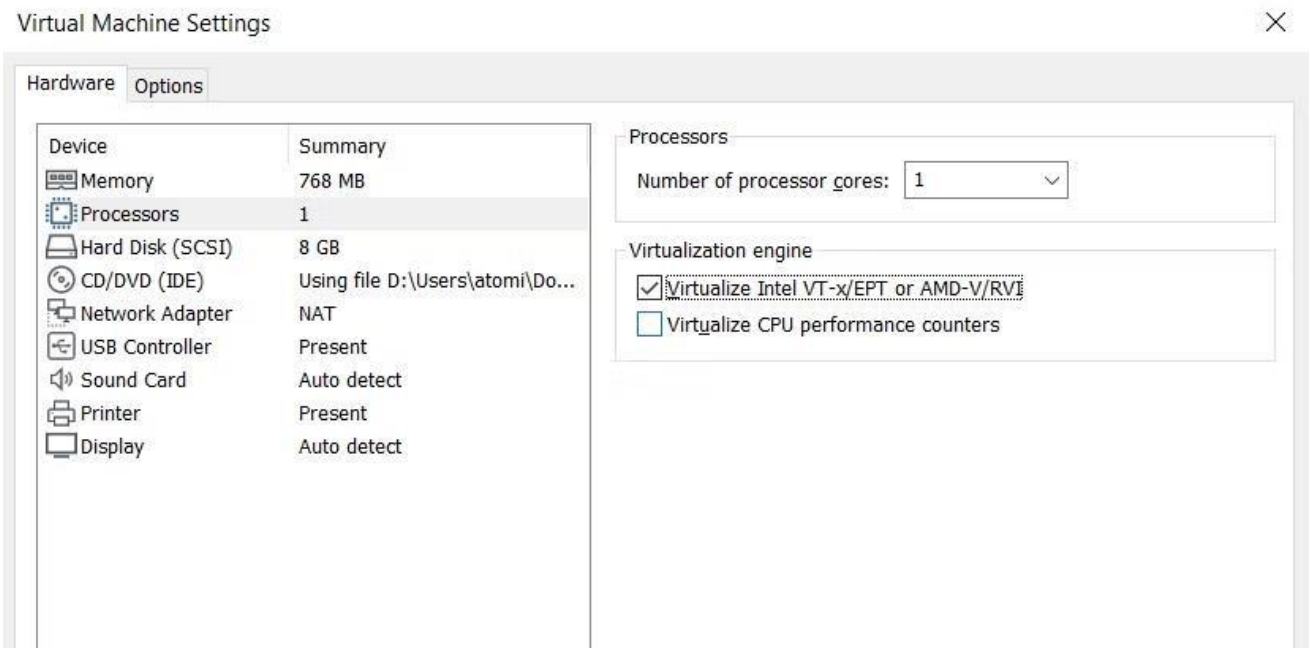


Your Linux virtual machine will be added to VMware Workstation Player.

Customize Your Virtual Hardware

In some cases, you might need to customize the virtual machine before installing Linux. Alternatively, you might install the OS and find there is something missing.

To fix this, right-click your virtual machine in VMware Workstation Player and select **Settings**.



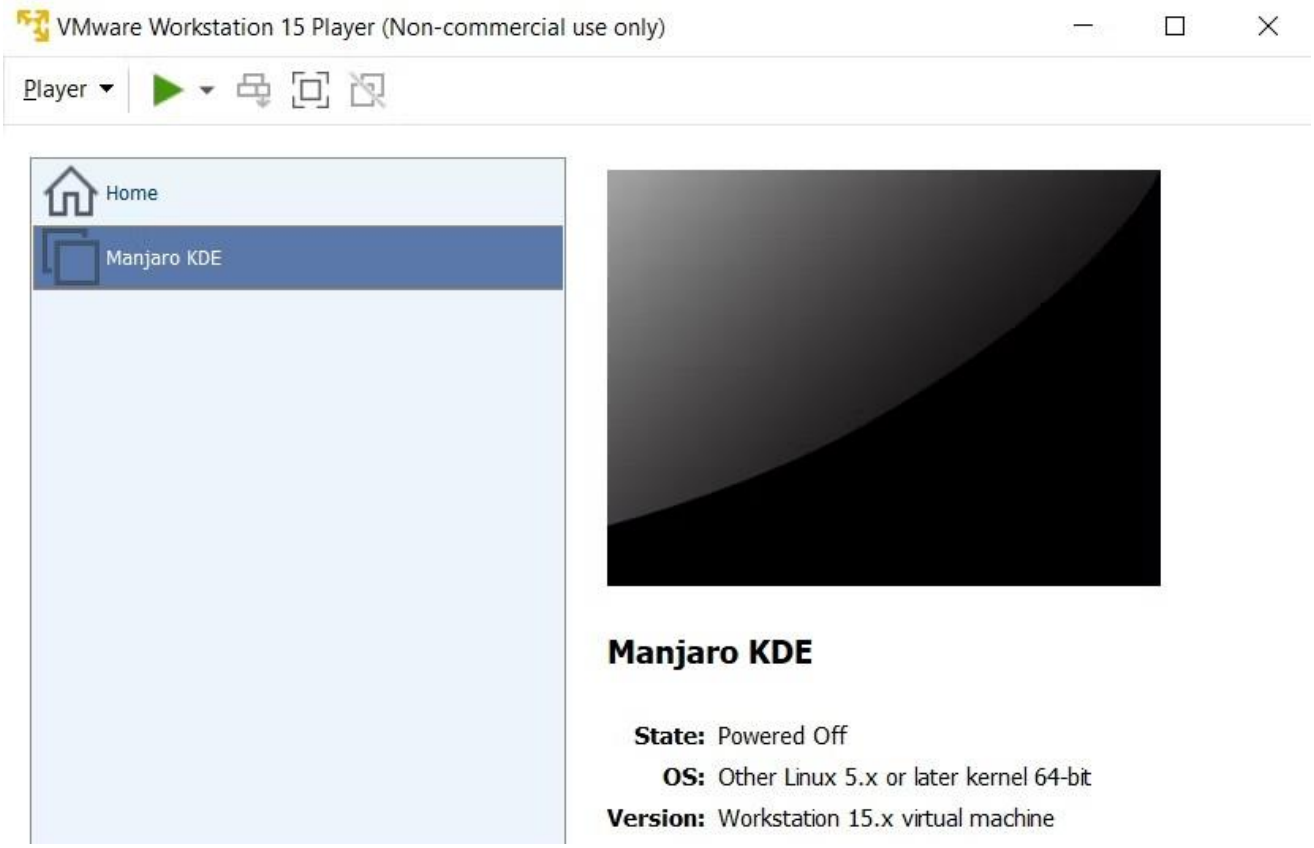
Here, you can tweak the virtual machine's hardware in other ways beyond the HDD. You have options for the **Memory**, **Processors**, **Network Adaptor** configuration, and much more.

It's worth taking a look at the **Processors** screen. In the right-hand pane, you'll spot a reference to a **Virtualization engine**. By default, this works automatically, but for troubleshooting set Intel VT-x or AMD-V, depending on your CPU.

You can address performance issues in the **Memory** screen. Here you'll spot an illustration of the suggested RAM size, as well as recommended options for your virtual machine. It's a good idea to stick to these recommendations. Going too small will prove a problem, while setting the RAM too high will impact on your PC's performance, slowing everything from standard system tasks to running the VM software!

Finally, spare a moment to check the **Display** settings. Default settings should be fine but if there is an issue with the display you can toggle 3D acceleration. Multiple monitors can be used and custom resolution set, but note that some modes will clash with some desktops.

Click **OK** to confirm changes, then select the virtual machine and click the **Play** button to begin.



Result: