PROGRAM - 1

AIM - Merge two sorted arrays and store in a third array

Step 1	Start
Step 2	Declare the variables
Step 3	Read the size of first array
Step 4	Read elements of first array in sorted array
Step 5	Read the size of second array
Step 6	Read the elements of second array in Sorted order
Step 7	Repeat step 8 and 9 while $i<m$ & $j<n$
Step 8	check if $a[i]>=b[j]$ then $c[k++]=b[j++]$
Step 9	Else $c[k++]=a[i++]$
Step 10	Repeat step 11 while $i<m$
Step 11	$c[k++]=b[j++]$
Step 12	Repeat step 13 while $j<n$
Step 13	$c[k++]=b[j++]$
Step 14	Print the First Array
Step 15	Print the Second Array
Step 16	Print the Merged Array
Step 17	End

PROGRAM - 2

AIM - Singly linked Stack - Push, pop, Linear search

Step 1 : Start
Step 2 : Declare the node and the required variables
Step 3 : Declare the functions for push, pop, display and search an element
Step 4 : Read the choice from the user
Step 5 : If the user choose to push an element, then read the element to be pushed & Call the function to push the element by passing the value to the function
Step 5.1 : Declare the newnode & allocate memory for the newnode
Step 5.2 - Set newNode → data = value
Step 5.3 - check if top == null then set newnode next = null
Step 5.4 - Set newnode → next = top
Step 5.5 - Set top = newNode & then print Insertion is successful
Step 6 : If user choose to pop an element from the stack then call the function to pop the element -

Step 6.1 - check if top == Null then print stack is empty

Step 6.2 - Else declare a pointer variable temp and initialize it to top

Step 6.3 - Print the element that being deleted

Step 6.4 - set temp = temp → next

Step 6.5 - free the temp

Step 7 - If the user choose the display then call the function to display the element in stack

Step 7.1 - check if top == Null then print stack is empty

Step 7.2 - Else declare a pointer variable temp and initalize it to top

Step 7.3 - Print temp → data

Step 7.4 - set temp = temp → next

Step 8 - If the user choose to search an element from the Stack then call the function to search an element.

Step 9 - End.

PROGRAM-3

AIM - Circular Queue.

Step 1 : Start

Step 2 : Declare the queue and other variables

Step 3 : Declare the functions for enqueue, dequeue search and display

Step 4 : Read the choice from the user

Step 5 : If the user choose the choice enqueue then Read the element to be inserted from the user and call the enqueue function by passing the value

Step 6 : If the user choice is the option dequeue then call the function dequeue

Step 7 : If the user choice is to display the queue then call the function display

Step 8 : If the user choice the search the call the function to search an element in the queue

Step 9 : End

PROGRAM - 4

AIM - Doubly linked list

Step 1 : start

Step 2 : Declare a structure and related variables.

Step 3 : Declare functions to create a node, insert a node in the begining, at the end and given position, display the list and search an element in the list

Step 4 : Define function to create a node, declare the required variables

Step 5 : Read the choice from the user to perform different operation on the list

Step 6 : If the user choose to perform insertion operation at the beginning then call the function to perform insertion

Step 7 : If the user choice is to perform insertion at the end of the list, then Call the function to perform the insertion at the end.

Step 8 : If the user choose to perform Insertion in the list at any position then Call the function to perform the Insertion operation

Step 9 : If the user choose to perform deletion operation is the list then call the function to perform the deletion operation

Step 10 : If the user choose to perform the search operation then Call the function to perform search operations

Step 11 : End

PROGRAM-5

AIM- Set Data Structure and set operations

Step 1 : Start
Step 2 : Declare the necessary variable.
Step 3 - Read the choice from the user to perform set operation
Step 4 - If the user choose to perform union
Step 4.1 - Read the cardinality of 2 sets.
Step 4.2 - check if $m_1=n$ then print cannot perform union
Step 4.3 : else read the elements in both the sets
Step 4.4 - Repeat the step 4.5 to 4.7 until i<m
Step 4.5 - C[i] = A[i]B[i]
Step 4.6 - Print C[i]
Step 4.7 - Increment i by 1
Step 5 - Read the choice from the user to Perform Intersection
Step 5.1 - Read the cardinality of 2 sets
Step 5.2 - check if m1=n then print Cannot perform Intersection
Step 5.3 - Else read the elements is both the sets
Step 5.4 - Repeat the step 5.5 to 5.7 until i<m
Step 5.5 - C[i]= A[i] & B[j]
Step 5.6 - Print C[i]

Step 5.7 - Increment i by 1

Step 6 - If the user choose to perform set difference operation

Step 6.1 - Read the cardinality of 2 sets.

Step 6.2 - check if m1=n then print Cannot perform Set difference operation

Step 6.3 - Else read the element in both sets

Step 6.4 - Repeat the step 6.5 to 6.6 until i<n

Step 6.5 - check if A[i]==0 then C[i]=0

Step 6.6 - Else C[i]=1

Step 6.7 - Increment i by 1

Step 7 ÷ Repeat the step 7.1 and 7.2 until i<m

Step 7.1 - print C[i]

Step 7.2 - Increment i by 1

Step 8 - End

PROGRAM - 6

AIM — Binary search Tree.

Step 1 : Start

Step 2 : Declare a structure and structure pointers for insertion deletion and search operations and also declare a function for inorder traversal

Step 3 : Declare a pointer as root and also the required variable.

Step 4 : Read the choice from the user to perform insertion, deletion, searching and inorder traversal.

Step 5 : If the user choose to perform insertion operation then read the value which is to be inserted to the tree from the user.

step 5.1 - pass the value to the insert pointer and also the root pointer.

step 5.2 - check if !root then allocate memory for the root.

Step 5.3 - Set the value to the Info part of the root and then allocate memory and return root.

Step 5.4 - check if root → Info > n then call the Insert pointer to Insert to left Of the root.

Step 5.5 - check if root → Info < n then call the Insert pointer to Insert to the left of the root.

step 5.6 Return the root.

step 6 - If the user choose to perform deletion operation then read the element to be deleted from the tree

Step 6.1 - check if not ptr then point node not found

Step 6.2 - Else if ptr → Info < n then call delete pointer by passing the right pointer and the item

Step 6.3 - else if ptr → Info > n then call delete Pointer by passing the left pointer and the Item

Step 6.4 - check if ptr → Info == item then check if ptr → left == ptr → right

then free ptr and return null

Step 7 : If the user choose to perform search operation the call the pointer to perform Search operation

Step 8 : If the user choose to perform traversal then call the traversal function and Pass the root pointers

Step 9 : End

PROGRAM-7

AIM - Disjoint set operations

Step 1 - Start
step 2 - Declare the structure and related variable
Step 3 - Declare function makeset()
Step 4 - Declare function display set.
Step 5 - Declare a function find and pass x to the function.
step 6 - Declare a function union & pass 2 variable x and y
Step 7 - Read the number of elements.
Step 8 - Call the function makeset.
Step 9 - Read the choice from user to perform union, find and display operation
Step 10 - If the user choose to perform union operation read the element to perform union and then call the function to perform union operation
Step 11 - If the user choose to perform find operation read the element to check if connected

Step 12 - else point Not Connected Component

Step 13 - the user choose to perform display operation then call the display set function

Step 14 - End.

Program - 8

AIM - Graph Traversal techniques DFS (using stack)

CODE

Step1 - Start

Step2 - Create a stack of nodes and visited array

Step3 - Insert the root in the stack

step4 - Run a loop till the stack is not empty

step 5 - Pop the element from the stack is not empty

step 6 - For every adjacent and unvisited node of current node, mark the node and insert it in the stack

step 7 - End