Program 2:

Aim: To merge two sorted arrays

Algorithm:

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 order.

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[i] then c[k++] = b[j++]

Step 9: Else c[k++] = a[j++]

Step 10: Repeat Step11 while j<n

Step 11: c[k++] = a[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 messaged array

step 17: stop

output

Size of first array
2
Enter value in sorted order
2
7
Size of second array
2
Enter value in sorted order
11
12
Array A
2
7

Array B
11
12

Merged array
5
9
10
17

program 2

Aim: To perform stack operations

Algorithm:
Step1 : start
Step2 : Declare the node and the required variables.
step3 : Declare the functions for push, pop, display and search an element.
Step4 : Read the choice from the user
Step5 : if the user choose to push an element. Then read the element to be pushed and call the function to push the element by passing the value to the function.
Step 5.1 : Declare the newnode and 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 = new node and Then print insertion is sucessfull.
Step 6 : if user choose to top 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 intialize 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 the stack.
Step 7.1: check it top = NULL then print stack is empty
Step 7.2: Else declare a pointer variable temp and initalize it to top
step 7.3: Repeat steps below while temp → next != null
Step 7.4: print temp → data
step 7.5: 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 8.1: Declare a pointer variable ptr and other necessary variable.
step 8.2: initialize ptr = top
Step 8.3: check if ptr = null then print stack empty.
Step 8.4: Else read the element to be searched.
step 8.5: Repeat 8.6 and 8.8 while ptr != null
set
Step 8.6: check if ptr → data == item then print element founded and to be located and set flag = 1
Step 8.7 Else set flag = 0
step 8.8 check if flag = 0 then print the element not found.
step 9. stop.

Output:

```
Menu
1. Push
2. POP
3. display
4. Seagch
5. Exit

enter The choice : 1
Enter The element to be inserted : 2
insertion is successful
Menu
1. Push
2. POP
3. display
4. search
5. Exit.
enter The choice : 1
Enter The element to be inserted : 4
insertion is successful.
Menu
1. Push
2. POP
3. display
4. Seagch
5. Exit
enter The choice : 1
enter The element to be inserted : 10
Insertion is sucessfull.
menu:
1. Push
2. POP
```

3. display
4. search
5. exist
enter the choice : 2
element deleted : 10

menu
1. Push
2. POP
3. display
4. search
5. exist
enter the choice : 3
4 → 2 → Null.

Program 3

Aim: To perform circular queue operations

Algorithm

Step 1 : Start
Step 2 : Declare the queue and the 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 5.1 : check in front == -1 and rear == -1 then set front = 0, rear = 0 and set queue [rear] = element
Step 5.2 : Else if rear + 1 % max == front or front = rear + 1 then print queue is overflow
Step 5.3 : Else set rear = rear + 1 % max and set queue [rear] = element.
Step 6 : if the user choice is the option dequee then call the function dequeue.
Step 6.1 : check if front == -1 and rear == -1 then print queue is underflow
Step 6.2 : Else check if front == rear then print the element is to be deleted then set front = -1 and rear = -1
Step 6.3 : Else print the element to be dequed set front = front + 1% max
Step 7 : if the user choice is to display the queue than call the function display.

step 7.1: check if front = -1 and rear = -1
Then print queue is empty.
Step 7.2: Else repeat the step 7.3 while
i <= rear
step 7.3: print queue[i] and set i=i+1.

Step 8: if the user choose the search then can the function to search an element in the queue.
Step 8.1: Read the element to be searched in the queue
step 8.2: check if item == queue[i] then print item found and its position and increment by 1
Step 8.3: check if c==0 then print item not found.
step 9: stop.

output
menu
1. insert
2. Delete
3. Display
4. Search
5. exist
enter the choice: 1
enter the number to insert: 10
menu
1. insert
2. Delete
3. display
4. Search
5. exist
enter the choice: 1
enter the number to insert: 20
menu
1. insert
2. delete
3. Display
4. Search
5. exist
enter the choice: 1
enter the number to insert: 30
menu
1. insert
2. Delete
3. display
4. search
5. exist
enter the choice: 3

10, 20, 30

Menu
1. insert
2. Delete
3. display
4. Search
5. exist
enter The choice : 4
enter The element which is to be searched : 30
item found at location 3:
menu
1. insert
2. Delete
3. display
4. search
5. exist
enter The choices 2.
10 was deleted

Menu
1. insert
2. Delete
3. display
4. Search
5. exist
enter The choice : 5.

program: 4

Aim: To perform doubly linked list operations:

Algorithm:
Step 2: start
Step2: Declare a structure and related variables
Step3: Declare functions to create a node, insert a node in the beigining at the end and given position display the ~~firs~~ list and search and element in the list.
Step 4: Define function to create a node, declare the required variables.
Step 4:1 set memory allocated to the node = temp then set temp → prev = null and temp → next = null
Step 4:2: Read the value to be inserted to the node.
Step 4:3: set temp→n = data and increment count by 1.
step 5: Read the choice from the uses to perform different operation on the list. at the begining then call the information to perm the insertion.
step 6.1: check if head == null then call the function to create a node, perform step 4:3
Step 6.2: set head = temp and temp1 = head.
Step 6.3 Else call the function to create a node, perform step 4 to 6.3 then set temp → next = head, set head → prev = temp and head = temp.

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 7.1 check if head == null then call the function to create a new node then set temp = head and then set head = temp2

Step 7.2 : else call the function to create a new node then set temp1 → next = temp temp → prev = temp1 and temp1 = temp

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 8.1 : Declare the necessary variables

Step 8.2 : Read the position where the node need to be inserted, set temp 2 = head

Step 9: if the user choose to perform deletion operation is the list then all the function to perform the deletion operation.

Step 9.1 : Declare the necessary variables

step 9.2 : Read the position where node need to be deleted set temp 2 = head

Step 9.3 : check if pos < 1 or pos >= count + 1 then print position out of range.

Step 9.4 : check if head == null the print the list is empty

Step 9.5 : write i < pos then temp 2 = temp 2 → next and increment i by 1.

Step 10: → if the user choose to perform the display operation then all the function to display the list.

Date: / /

Step 10.1 : set temp 2 = 2
Step 10.2 : check if temp 2 = null then print list is empty.
Step 11 : if the user choose to perform the search operation then call the function to perform search operations

Step 11.1 : Declare the nessary variables.
Step 11.2 step temp 2 = head
Step 11.3 : check if temp 2 = = null then print the list is empty.
Step 11.4 : Read the value to be searched.
Step 11.5 : while temp 2 : = null then check if temp 2 → = = data then print element found at position count + 1
Step 11.6 : else set temp 2 = temp 2 → next and increment count by 1
Step 11.7 : Print element not found in the list

Step 12 : stop.

Output:

1. insert at beginning
2. insert at end
3. insert at position
4. Delete
5. Display
6. Search
7. exist

enter choice :1

enter the value to node :5

1. insert at beginning
2. insert at end
3. insert at position
4. Delete
5. Display
6. search
7. exist

enter choice 1

enter the value to node :10

1. insert at begining
2. insert at end.
3. insert at position
4. Delete
5. Display
6. Search
7. exist.

enter choice :2

enter value to node 2

program 5

Aim: To perform set operations

Algorithm:
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 ~~else~~ if m,=n then print cannot perform union
step 4:3: check 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 4:6: increment i by 1
step 5: Read the choice from the user to perform inseration.
step 5:1: Read the cardinality of 2 sets.
step 5:2: check if m=n then print cannot perform intersection
step 5:3: else read the elements is both the ~~the~~ sets.
step 5:4: Repeat the step 5:6 to 5:7 until i<m
step 5:5 : C[i] = A[i] & B[i]
step 5:6 = print C[i]
step 5:7 = increcrement i by 1
step 6: if the user choose to operform set differentiate operation.

Step 6.1 : Read the cardinality of a sets
step 6.2: check if $m_1 = n$ 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 6.8 until $i<m$
Step 6.5: check if $A[i] == 0$ Then $c[i] = 0$
step 6.6: Else if $B[i] == 1$ Then $c[i] = 0$
step 6.7: Else if $B[i] == 1$ Then $C[i] = 0$
step 6.8: increment i by 1
Step 7: Repeat step 7.1 and 7.2 until $i<m$
Step 7.1: print $C[i]$
Step 7.2: increment i by 1.

Output:

press 1 for union
press 2 for intersection
press 3 for subtraction
press 4 for exist
enter choice
enter the size of set 1
3
enter the elements of set 1
1
2
3
enter the size of set 2
2
enter the elements of set 2
2
3
union: 1, 2, 3

Program : 6

Aim: To perform Binary search tree operations

Algorithm

Step 1: start

step2: 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.

step4: Read the choice from the user to perform insertion, deletion, searching and inorder traversal

step5: if the user chose 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 into part of the root and then set left and right part of the root to null and return root.

Step 5.4: check if root → info > x then call the insert pointer to insert to left of the root

Step 5.6: Return to boot.

Step 6: if the user choose to perform deletion operation then read the element to be deleted from trace pass the root pointer and the then to the delete pointer.

Step 6.1: check if not ptr then print node not found.

Step 6.2: else if ptr→info < x then call delete pointer by passing the right pointer and the item

Step 6.3: else if ptr→info > x then call dete 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 then call the pointer to perform search operation.

step 7.1: Declare the nessary pointers and variables

step 7.2 Read the elements to be searched.

step 7.3 while ptr check if item > ptr→info then ptr = ptr→right.

step 8: if the user choose to perform traversal then call the traversal function and pass the root pointers.

step 8.1: if root not equals to null recursive call the functions by passing root→left

Step 8.2: print root → info

Step 8.3: call the traversal function recursively by passing root → right

Step 9: end

Output:

1. insert in Binary tree
2. Delete from Binary tree
3. In order traversal of Binary tree
4. Search
5. exist

enter choice : 1
enter new element : 20
root is 20
In order traversal of binary tree is : 20

1. Insert in Binary tree
2. Delete from Binary tree
3. Inorder traversal of Binary tree
4. search
5. exist

enter choice : 1
enter new element : 25
inorder traversal of binary tree is 20, 25

program 7

Aim: To perform Disjoint set operations

Algorithm:

step1 : start
step2: declare the structure and related variable
step3: Declare function makeset()
step 3.1 : Repeat step 3.2 to 3.4 until i<n
step 3.2 : disparent [i] is set to 1
step 3.3 : set distank [i] is equal to 0
step 3.4: Increment i by 1
step 4 : Declare a function display i<n
step 4.2 : print disparent [i]
step 4.3: increment i by 1
step 4.4: Repeat step 4.5 and 4.6 until i<n
step 4.5 : print dis rank [i]
step 4.6 : increment i by 1
step 5: Declare a function find and pass x to the function.
step 5.1 : check if disparent [n] = x then set the return value to disparent [x]
step 5.2 : return disparent [x]
step 6 : Declare a function union and pass to variables x and y.
step 6.1 set x set to find (x)
step 6.2 : set y set to find (y)
step 6.3 check if y set = = x set then return
step 7: Read the number of elements
step 8 : Call the functions masked.
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 11.1: check if find(x) == find(y) then print connected component.

Step 11.2: else print Not Connected component.

Step 12: if the user choose to perform display operation then call the display set function.

output:

How many elements ? 4

menu
1. union
2. find
3. display
enter choice 1
enter elements to perform union:
3
4
Do you wish to continue ? (1/0)
1
menu
1. union
2. find
3. Display
enter choice 1
enter elements to perfor union
5
6
Do you wish to continue ? (1, 0)
0