PROGRAM - 1

AIM :

To Merge two sorted arrays and store in a third array.

ALGORITHM :

Step 1 : Start.

Step 2 : Declare the variable.

Step 3 : Input the elements in first and second array.

Step 4 : Repeat step 5 and 6 while i<m and j<n

Step 5 : check if ar1[i] >= ar2[j], then
ar3(k++] = ar2[j++].

Step 6 : Else ar3[k++] = ar1[i++].

Step 7 : Repeat step 8 while i<m.

Step 8 : ar3[k++] = ar1[i++].

Step 9 : Repeat step 10 while j<n.

Step 10 : ar3[k++] = ar2[j++].

Step 11 : Display the first array

Step 12 : Display the second array.

Step 13 : Display the New Merged array.

Step 14 : End.

OUTPUT :

Enter the number of elements in array1 : 4.

Enter elements in array1 : 12 34 56 78.

Enter the number of elements in array2 : 5

Enter elements in array2 : 33 43 52 86 90

First array :

12
34
56
78.

Second array :
33
43
52
86
90

New Merged array

12
33
34
43
52
56
78
86
90

PROGRAM - 2

AIM:

To implement circular queue and perform add, delete, and search operations.

ALGORITHM:

Initializing queue [MAX].

Set front = -1 and rear = -1

Algorithm to insert element:

Step 1: if (rear+1) % MAX = front

Print "overflow".

Goto step 4.

[End if]

Step 2 : if front = -1 and rear = -1.

set front = rear = 0

else if rear = MAX-1 and front != 0.

set rear = 0

else

set rear = (rear+1 %) % MAX.

[End if]

Step 3 : Set queue[rear] = val

Step 4 : Exit.

Algorithms to delete element.

Step 1 : if front = -1
print "underflow"
Goto step 4
[End if]

Step 2 : set val = queue[front]

Step 3 : if front = rear.
Set front = rear = -1
else
if front = MAX - 1
set front = 0.
else.
set front = front + 1.
[End if].
[End if]

Step 4 : Exit.

OUTPUT :

1. Insertion
2. Deletion
3. Display
4. Search
5. Exit

Enter your choice : 1
Enter the element which is to be inserted : 10

1. Insertion
2. Deletion
3. Deletion Display
4. Search .
5. Exit

Enter your choice : 1
Enter the element which is to be inserted : 20

1. Insertion
2. Deletion
3. Display
4. Search
5. Exit

Enter your choice : 1
Enter the element which is to be inserted : 30

1. Insertion.
2. Deletion
3. Display
4. Search.
5. Exit

Enter your choice : 2.

the dequed element is 10.

1. Insertion.
2. Deletion
3. Display
4. Search.
5. Exit.

Enter your choice : 3.

Elements in a Queue : 20 30

1. Insertion
2. Deletion.
3. Display
4. Search.
5. Exit

Enter your choice : 4.
Enter the element which is to be search : 20
Item found at location 2.

PROGRAM 3.

AIM :

To implement search singly linked stack and perform push, pop and linear search.

ALGORITHM :

Step 1 : start.

Step 2 : Print menu. 1. Push 2. Pop . 3) Display .
step 4) Search 5) Exit.

Step 3 : If choice is 1, if no go ~~go~~ to step 4
a) Read the element to be inserted.
b) Create a node with this number.
3) Insert the node after header node, go to step 2 .

Step 4 : If the choice is 2, if no go to step 5 .
a) Change the pointer of header node to

next of the node to which it already points.

b) Delete the node that was previously next of header node to which it already point.

c) If next of header is NULL, then print stack is empty.

Step 5) If the choice is 3, if no goto step 6.

a) traverse the list from the header and print the data part of each node

Step 6) If the choice is 4, if no got to step 7.

a) search the element that you want to be search in the list

Step 7) Exit from program

Step 8) Stop

OUTPUT.

1. PUSH
2. POP
3. DISPLAY
4. LINEAR SEARCH.
5. EXIT.

Enter your choice : 1
Enter the value : 10
Value pushed.

1. PUSH
2. POP
3. LINEAR SEARCH.
4. DISPLAY
5. EXIT.

Enter your choice : 1
Enter you the value : 12

1. PUSH
2. POP
3. LINEAR SEARCH
4. DISPLAY
5. EXIT

Enter your choice : 2
Value Deleted.

1. PUSH.
2. POP
3. LINEAR SEARCH
4. DISPLAY
5. EXIT.

Enter your choice : 4.

10->NULL.

Ente

1. PUSH
2. POP
3. LINEAR SEARCH.
4. DISPLAY
5. EXIT.

Enter your choice : 3.

Enter the element to search : 10.

Element found 10

1. PUSH
2. POP
3. LINEAR SEARCH.
4. DISPLAY
5. EXIT.

Enter your choice : 5

PROGRAM - 4

AIM:

To implement doubly linked list and perform insertion, deletion and search.

ALGORITHM:

Step 1: Start.

Step 2: Declare a structure and related variables

Step 3: Declare functions to create a node, insert a node in the beginning, 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 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.

If the choice is insertion at beginning

Step 5 : check if head == null,
then call the function to create a node, perform step 4 to 4.3.

Step 5.1 : Set head = temp and temp1 = head

Step 5.2 : Perform step 4 to 4.3 then set temp -> next = head, set head -> prev = temp and head = temp

If the choice is insertion at end.

Step 6 : check if head == null then,
call the function to perform the insertion at end.

Step 6.1 : Else call function to create a new node then set temp -> next = temp, temp -> prev = temp1 and.
temp1 = temp

If the choice is insertion at any position.

Step 7 : Declare the variables

Step 7.1 : Read the position where the node need to be inserted, set temp2 = head.

Step 7.2 : Check if pos<1 or pos> = count+1, then position is out of range

Step 7.3 : Check if head = =null and pos=1 then print "Empty list" cannot insert other.

Step 7.4 : Check set temp = head and head=temp1.

Step 7.5 : while i<pos then set,
set temp2 = temp2 -> next then increment i by 1.

Step 7.6 : Call the function to create new node then set temp -> prev = temp2. temp->next = temp2 -> next -> prev = temp.
temp2 -> next = temp.

If choice is to perform deletion operation.

Step 8 : Declare variables.

Step 8.1 : Check if pos <1 or pos >= count + 1.
print position out of ~~then~~ range.

Step 8.2 : Check if head == null, then.
print the list is empty.

Step 8.3 : while i<pos then
temp2 = temp2 -> next and increment
i by 1.

Step 8.4 : Check if i==1 then check if temp2 -> next == null then print node deleted free (temp 2).
set temp2 = head = null.

Step 8.5 : check if temp2 -> next == null then
temp2 -> prev -> next = null then
free (temp2) then print node deleted.

Step 8.6 : temp 2 -> next -> prev = temp 2 -> prev
then check if (i != 1 then temp2 ->
prev -> next = temp2 -> next.

Step 8.7 : check if i = 1 then head = temp2 → next then print node deleted then free temp2 and decremented count by 1.

If option is to ~~per~~ display operation.

Step 9 : Set temp2 = n.

Step 9.1 : check if temp2 = null = no then print list is empty.

Step 9.2 : while temp2 → next != null then print temp2 → n then temp2 = temp2 → next.

~~Step~~ If option is to perform search.

Step 10 : ~~free~~ Set temp2 = head

Step 10.1 : check if temp2 == null then print the list is empty.

Step 10.2 : Read the value to be searched.

Step 10.3 : while temp2 != null
check if temp2 → n == data then
print element found at position count + 1.

Step 10.4 : Else set temp2 = temp2 → next and increment

Step 10.5 : Print element not found in list.

Step 10.6 : End.

OUTPUT

Choose one option from the following list...

1. Insert in beginning
2. Insert at last
3. Insert at any random location
4. Delete from Beginning
5. Delete from last
6. Delet the node after the given data
7. Search
8) Show
9) Exit

Enter your choice?

1.

Enter Item value 12.

Node Inserted.

Choose one option from the following list...

1) Insert in Begining.
2) Insert at last
3) Insert at any random location.

4) Delete from beginning
5) Delete from last
6) Delete the node after the given date
7) Search
8) Show
9) Exit

Enter your choice ?
2.
Enter value 45
node inserted.

choose one option from the following list....

1. Insert in begining
2. Insert at last
3. Insert at any random location
4. Delete from Beginning.
5. Delete from last
6. Delete the node after the given date
7. Search
8) Show
9) Exit

Enter your choice?
3
Enter the location 1
enter value 11

node inserted
choose one option from the following
list...
1. Insert in beginning
2. Insert at last
3. Insert at any random location
4. Delete from beginning
5. Delete from last
6. Delete the node after the given data.
7. Search
8. Show
9. Exit
Enter your choice : 4.
node deleted.
choose one option from the following list..
1. Insert in beginning
2. Insert at last
3. Insert at any random location
4. Delete from beginning
5. Delete from last
6. Delete the node after the given data
7) Search
8) Show
9) Exit

Enter your choice
8.
printing values....
45
11
choose one option from the following list.
1. Insert in begining.
2. Insert at last
3. Insert at random location
4. Delete from Beginning.
5. Delete from last
6. Delete the node node after the given data.
7. Search
8) show.
9) exit
Enter your choice ?
5
node deleted
choose one option from the following list
1. Insert in begining
2. Insert at last
3. Insert at random location
4. Deletion from beginning
5. Delete from last.
6. Delete the node after the given the data

7) Search
8) Show
9) Exit

Enter your choice : 9.

PROGRAM - 5

AIM :

To implement Binary search trees and Perform Insertion, Deletion and search.

ALGORITHM :

Step 1 : Start.

Step 2 : Declare the necessary variables structure and structure pointers for insertion, deletion and search and also a function for inorder traversal.

Step 3 : Declare a pointer as root and variable

Step 4 : Read the choice.

Step 5 : If option is insert, then read the value which is to inserted to the tree from the user.

Step 5.1 : Pass the value to the insert pointer & 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 the set left & right part of the root to null & return root.

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

Step 5.5 : check if root → info > x then call the insert pointer to insert to the right of root.

Step 5.6 : Return the root.

Step 6 : If choice is deletion.

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

Step 6.2 : Else if pt → info < x then call delete pointer by passing the right pointer &item

Step 6.3 : Else if ptr → info > x then call delete pointer by passing the left pointer & item.

Step 6.4 : check if ptr → info == item then
check if ptr → left == ptr → right
then free ptr and return null.

Step 6.5 : Else if ptr → left == null then
set p1.ptr → right & free ptr, return p1

Step 6.6 : Else if ptr → right == null then
set p1. ptr → left & free ptr, return p1

Step 6.7 : Else set p1 = ptr → right & p2 = ptr → right

Step 6.8 : while p1 → left not equal to null,
set p1 → left . ptr → left & free ptr
return p2 .

Step 6.9 : Return ptr.

Step 7 : If option is search

Step 7.1 : Declare the necessary pointers & variables

Step 7.2 ~~Declare~~ Read the element to be
searched

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

Step 7.4 : Else if item < ptr -> info then
ptr = ptr -> left ..

Step 7.5 : Else break .

Step 7.6 : Check if ptr then print that the element is found .

Step 7.7 : Else print element not found in tree and return root .

Step 8 . If option is traversal, call traversal function and pass the root pointer .

Step 8.1 : if root not equals null recursively call the function by passing root -> left

Step 8:2 : print root -> info .

Step 8:3 : call the traversal function recursively by passing root -> right .

OUTPUT

1. Insertion in BST
2. Deletion in BST
3. Search element in BST.
4. Inorder Traversal
5. Exit.

Enter your choice 1
Enter your ~~data~~: 12
Continue Insertion (0|1) : 1
Enter your data : 66
Continue Insertion (0|1) : 1
Enter your data : 77
Continue Insertion (0|1) : 1
Enter your data : 88.
Continue Insertion (0|1) : 0

1) Insertion in BST.
2) Deletion in BST
3) Search Element in BST.
4) Inorder Traversal
5) Exit.

Enter your choice : 2.
Enter your data : 66.

1. Insertion in BST.
2. Deletion in BST
3. Search Element in BST.
4. Inorder Traversal
5. Exit.

Enter your choice : 3.
Enter value for data : 77
data found.

1. Insertion in BST.
2. Deletion in BST
3. Search Element in BST.
4. Inorder Traversal
5. Exit
Enter your choice : 5

PROGRAM – 6

AIM :

To perform set data structure and set operation (Union, Intersection & Difference Using Bit string.

ALGORITHM :

Step 1 : Start.

Step 2 : Declare the necessary variables.

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 != n, then print cannot perform union.

Step 4.3 : Else read the elements in both the sets.

Steps 4.4 : Repeat the steps 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 m != n then print cannot perform intersection

step 5.3: Else read the elements in 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[i]

step 5.6: print c[i]

step 5.7: increment i by 1.

step 6: If option is difference.

step 6.1: Read the cardinality of 2 set.

steps 6.2: check if m != n then print cannot perform set difference operation.

Step 6.3 : Else read the element in both set.

Step 6.4 : Repeat the step 6.5 to 6.8 until i<n.

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 c[i] = 1

Step 6.8 : Increment i by 1

Step 7 : Repeat the step 7.1 to 7.2 until i<n

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 exit .
Enter choice 1
Enter the size of set 1
3.
Enter the element of set 1.
1
2
3.
Enter the size of set 2 .
2
Enter the element of set 2 .
2
3.
Union : 1 2 3 .

P

Press 1 for union

press 2 for intersection

press 3 for subtraction

press 4 for exit

Enter your choice : 2 .

Enter the size of set 1.

3 .

Enter the element of set 1 .

1

2

3 .

Enter the size of set 2 .

2 .

Enter the element of set 2

3 .

4

~~differ~~ Intersection : 3 .

Press 1 for union

press 2 for intersection

press 3 for subtraction

press 4 for exit .

Enter the your choice 3.
Enter the size of set 1.
3
Enter the element of set 1.
1
2
3
Enter the size of set 2.
2.
Enter the element of set 2.
3
2.
difference 4.

Press 1 for union
Press 2 for intersection.
Press 3 for subtraction.
Press 4 for exit.
Enter your choice : 4.

PROGRAM - 7.

AIM :

To perform disjoint sets & the associated operation (create, union, find).

Step 1 : Start.

Step 2 : Declare the structure and related variables.

Step 3 : Declare function makeset()

Step 3.1 : Repeat step 3.2 to 3.4 until i<n.

Step 3.2 : dis parent [i] is set to i

Step 3.3 : set dis rank [i] is equal too

Step 3.4 : Increment i by 1.

Step 4 : Declare a function display set.

Step 4.1 : Repeat step 4.2 to 4.3 until i<n

Step 4.2 : print dis parent [i]

Step 4.3 : Increment i by 1.

Step 4.4 : Repeat step 4.5 to 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 & pass x to the function.

Step 5.1 : check if disparent [x] != x then set the return value to disparent [x].

Step 5.2 : return disparent [x]

Step 6 : Declare a function union & pass 2 variables x & y.

Step 6.1 : set xset to find(x).

Step 6.2 : set yset to find(y).

Step 6.3 : check if yset == xset then return

Step 6.4 : set ~~xyset~~ check if disrank [xset] < disrank [yset] then.

Step 6.5 : set yset = disparent [yset]

Step 6.6 : set -1 to disrank [xset]

Step 6.7 : else if check disrank [xset] > dis rank [yset]

Step 6.8 : set xset to disparent [yset].

Step 6.9 : Set -1 to dis.rank[Yset]

Step 6.10 : else dis.parent[Yset] = xset

Step 6.11 : set disrank[xset] + 1 to disrank[xset]

Step 6.12 : set -1 to disrank [Yset]

Step 7 : Read the no. 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 & 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 11.2 : else print not connected component

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

Step 13 : End .

Output:

How many element ?. 4.

Menu.

1. Union
2. End
3. Display

Enter choice :

1.

Enter elements to perform union :

3
4

Do you want to continue ?. (1/0)

1.

Menu.

1. Union
2. End
3. Display

Enter choice :

1.

Enter element to perform union :

5
6

Do you wish to continue? (1/0).

1.

Menu-

1. Union

2. Find

3. Display

Enter choice:

3

Parent Array

3 1 2 3

Rank array.

-1 0 0 1

Do you wish to continue? (1/0).