Program 1

Aim: To Merge two Sorted arrays and Store in a third array.

ALGORITHM:

Step1 : Start

Step2: Declare the variable

Step3: Input the element in first and Second Array

Step4: Repeat Step 5 and 6 while i<m and j<n

Step 5: Check if $a_1[i] >= a_2[j]$, then

$$a_3[k++] = a_2[j++]$$

Step 6: else $a_3[k++] = a_1[i++]$

Step 7: Repeat Step8 while i<m

Step8: $a_3[k++] = a_1[i++]$

Step 9: Repeat Step 10 while j<n

Step10: $a_3[k++] = a_2[j++]$

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 the elements in Array1 : 12 34 56 78

Enter the number of elements in

Array2 : 5

Enter the 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

Initilize queue [MAX]

Set front = -1 and rear = -1

Algorithm to insert element.

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

Print "Overflow"

go to Step 4

[End If]

Step 2: If front = -1 and rear = -1

Set front = rear = 0

elseif rear = MAX - 1 and front != 0

Set rear = 0

else

Set rear = (rear+1) % MAX

[End If]

Step3: Set queue[rear] = val

Step4: Exit

Algorithm for delete element

Step1: If front = -1

Print "Overflow"

Goto Step4

[End If]

Step2: Set val = queue[front]

Step3: If Front = rear

Set front = rear = -1

else

If front = MAX-1

Set front = 0

else

Set front = front+1

[End If]

[End If]

Step4: Exit

OUTPUT

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

Enter your choice : 1

Enter the elements which is to be

Inserted: 10

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

Enter your choice : 1

Enter the elements 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 queue is 20 30
1. Insertion
2. Deletion
3. Display
4: Search
5: Exit
Enter your Choice: 4
Enter the element which is to be Searched: 20
item found of location 20

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 4. Search 5. exit

Step 3: If Choice is 1, If no go to Step 4

a) Read the element to be inserted

b) Create a node with their number

c) Inserted the node after header node, goto 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 Points

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

Steps: If the Choice is 3, If no goto Step6.

a) Traverse the list from the header and Print the data part of each node.

Step6: If the Choice is 4, If no goto 7

a) Search the elements that you want to be Search in the list

Step7: Exit from Program

Step8: 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. display
4. linear Search
5. exit

Enter your Choice: 1

Enter 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

1. Push
2. Pop
3. linear Search
4. display
5. exit

Enter your choice : 5

Enter the element for Search 10

element found 10.

1. Push
2. Pop
3. linear Search
4. display
5. exit

Enter your choice: 5

Program 4

Step1 : Start

Step2: Declare a structure and related variables

Step3: Declare function to create a node, insert a node in the beginning, at the end and given position, display the ~~beginning, at the~~ list and search an element in the list.

Step4: Define function of create a node, declare the required variables.

Step4.1 : Set memory allocated to the node = temp, then set temp->Prev=NULL and temp->next = null.

Step4.2: Read the value to be inserted to the node

Step4.3: Set temp -> n = data and increment count by 1

Step5: Read the Choice from the User to Perform different Operation on the list.

Step 6: If the user chooses to perform insertion Operation at the beginning, then all the function to Perform the insersion.

Step6:1: Check of head == NULL then Call the function to Create a node, Perform Steps 4 to 4:3

Step 6:2: Set head = temp and temp1 = head.

Step6:3: else Call the function to Create a node. Perform Step 4 to 4:3 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

Create a new node then Set temp = head and then set head = temp1.

Step 7:2: else call the function to create a new node then set temp1 -> next = temp, temp -> prev = temp1 and temp1 = temp

Step8: If the user chooses to perform insertion in the list at any position then call the function to perform the insersion Operation.

Step 8:1: Declare the neccesary variable

Step 8:2: Read the Position where the node needs to be inserted. Set temp2 = head.

Step8:3: Check if pos<1 or pos > = Count+1 then print the position is out of range.

Step8:4: check if head == null and pos = 1 then print "empty list cannot insertes other than 1st position"

Step 8:5: Check if head == null and pos=1 then call the function to create newnode, then set temp = head and head = temp.

Step 8:6: while i<pos then set temp = temp -> next then increment i by 1.

Step 8:7: Call the function to create a new node and then set temp -> prev = temp3 temp -> next = temp2 -> next -> Prev = temp, temp2 -> next = temp

Step 9: If the user chooses to perform deletion operation in the list then call the function to perform the deletion operation.

Step 9:1 Declare the neccessary variables

Step 9:2: Read the position where node needs to be deleted, Set temp2 = 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 then print the list is empty.

Step 9:5: while i < Pos then Set temp2= temp2 -> next and increment i by 1.

Step 9:6: check if i == -1 then check if temp2 -> next == null then print node deleted free (temp2) Set temp2 = head = null

Step 9:7: check if temp 2 -> next == null then temp2 -> prev -> next = null then free (temp2) then print node deleted

Step 9:8: temp2 -> next -> Prev = temp -> next then print node deleted then free temp2 and decrement count by 1.

Step 9:9: check if i == 1 then head = temp2 -> next then print node deleted then free temp2 and decrement

count by 1

Step 10: If the user choosen to Perform the display operation then call the function to display the list

Step 10:1 : Set temp 2 = h

Step 10:2 : Check if temp2 == null then Print list is Empty.

Step 10:3 : while temp2 -> next1 = null then Print temp2 -> n then temp2 = temp2 -> next.

Step 11: If the user choosen to perform the Search Operation then call the function to perform Search operation.

Step 11:1 : Declare the neecessary variables

Step 11:2 : Set temp2 = head.

Step 11:3 : Check If temp2 == null then Print the list is empty

Step 11:4 : Read the value to be Searched.

Step 11:5 while temp2 != null then Check.
If temp2 -> n == data. then Print
element found at position Count+1.

Step 11:6 : else Set temp2 = temp2->next
end increment count by 1.

Step 11:7: Print element not found
in the list

Step 12: End.

OUTPUT

1. Insert at beginning
2. Insert at end
3. Insert at position 1
4. Delete at 1.
5. Display from beginning
6. Search for element.
7. exit

Enter choice: 1

Enter value to node: 1

Enter Choice: 1

Enter value to node: 2

Enter choice: 1

Enter value to node: 3

Enter choice: 2

Enter choice to node: 7

Enter choice: 5

linked list elements from beginning:

3 2 1 7

Program 5

Aim: Set data structure and set operations (Union, Intersection and difference) using bit string.

Step 1: Start

Step 2: Declare the neccessary variables

Step 3: Read 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 m1 =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 m!=n then print Can't Perform insertion.

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[i]

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 m!=n then Print

Cannot Perform Set difference Operation

Step 6:3 : else read the elements in both Sets.

Step 6:4 : Repeat the Step 6:5 to 6:8 Untill 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 and 7.2 Untill i<m

Step 7.1 : Print C[i]

Step 7.2 : increment i by 1

OUTPUT

Input Choice to Perform.

1. Union 2. Intersection 3. difference 4. exit

Enter the Cardinality of first Set: 3.

Enter the Cardinality of Second set: 3.

Enter element to first Set: (0/1) 1

0

1

Enter element of Second Set: (0/1) 0

1

0

Element of Set 1 Union Set 2 : 1 1 1

Program 6.

Aim: Binary Search tree - Insertion, deletion, Search.

Step1: Start

Step2: Declare a structure and structure Pointer for insertion, deletion and Search Operation and also declare a function for inorder traversal

Step3: 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 chooses 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: Setect the value to the Info 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 5.5: 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 the User choose to Perform deletion operation then read the element to be deleted from the tree pass the

root Pointer and the item to the delete Pointer.

Step6:1 : Check if not ptr then Print node not found

Step6:2: else if ptr->info.<x then Call delete pointers by passing the right Pointer and the item.

Step6:3. elseif ptr->info >x then Call delet Pointer by passing the left pointer and the item.

Step6:4. Check if ptr->info==item then check if ptr->left ==ptr->right then free ptr and return null.

Step6:5. Else if P1.ptr->left==null then Set ptr->right and free Ptr, return P1.

Step6:6. else if ptr->right==null, then Set P1.ptr->left and free ptr, return P1.

Step6:7: else set P1=ptr->right and P2=ptr->right

Step 6:8: while $P_1 \rightarrow$ left not equal to null, Set $P_1 \rightarrow$ left = ptr $\rightarrow$ left and free ptr, return P_2.

Step 6:9 : Return ptr

Step 7: If the User choose to Perform Search Operation then Call the Pointer to Perform Search Operation.

Step 7.1: Declare the neccessary pointers and variables

Step 7:2: 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 returns root.

Step8: If the user Choose to Perform traversal than call the traversal function and Pass the root pointers.

Step8.1 : If root not Equals to null recursively Call the functions by passing root -> left

Step 8:2: Print root -> info

Step 8:3: Call the traversal function recurvely by passing root -> right

OUTPUT

1. Insert in binary tree
2. Delete from binary tree
3. inorder traversal of Binary tree.
4. Search
5. exit.

enter choice : 1

enter new element : 50

Root is 50

inorder traversal of binary tree is: 50

1. Insert in binary tree
2. Delete from binary tree
3. inorder traversal of Binary tree
4. Search
5. exit.

enter choice: 1

enter new element : 25

Inorder traversal of binary tree is

: 25 50

Program-7

Aim: Disjoint set and the associated operations (Create, Union, find)

Step 1: Start

Step 2: Declare the structure and related structure variable

Step 3: Declare a 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 to 0.

Step 3.4: increment i by 1.

Step 4: Declare a function display Set

step 4.1: Repeat Step 4.2 and 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 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 dis.parent[n] != x then set the return value to dis.parent[n]

Step 5.2: Return dis.Parent[n].

Step 6: Declare a function union and pass two 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 xset == yset then return.

Step 6:4: Check if dis.rank [x set] <
dis.rank [y set]

Step 6:5 : Set y set = dis.parent [y set]

Step 6:6 : Set -1 to dis.rank [x set]

Step 6:7: else if check dis.rank [x set]
> dis.rank [y set]

Step 6:8: Set x set to dis.parent [y set]

Step 6:9: Set -1 to dis.rank [y set]

Step 6:10 else dis parent [y set] = x set

Step 6:11 : Set dis.rank [x set] + 1 to
dis rank [x set]

Step 6:12: Set -1 to dis.rank [y set]

Step 7: Read the number of elements

Step 8: Call the function make set.

Step 9: Read the choile from User to
perform union, find and display oper-
ation.

Step 10: If the users chooses to Perform Union operation, read the element to Perform Union, then Call the function to Perform Union Operation.

Step 11: If the user chooses to Perform find operation, read the element to check it Connected.

Step 11.1: check if find (x) == find (y) then Print Connected Components.

Step 11.2: else print not connected. Components

Step 12: If the users chooses to Perform display Operation call the function display Set.

Step 13: End.

OUTPUT

How many elements: 4

Menu
1. Union
2. find
3 display

Enter Choice: 1

Enter element to perform Union

3
4

Do you want to Continue (1/0): 1