Program 1

Aim: Program to Merge sorted arrays

Algorithm:-

Step 1: START

Step 2: Initialize arr [] arr2 [], res [], m, n, j, k

Step 3: Check while $i<m$ && $j<n$ and arr1 > arr2 [j] then set result,

res [k++] = arr2 [j++]

otherwise

arr1 [i++]

Step 4: Repeat while $i<m$ && $j<n$ and arr1 > arr2 then set res [k++] = arr1 [j++]

Set i = i + 1

else

Set k = k + 1

Step 5: Repeat while $j<n$ then set res [k++] = arr2 [j++]

Step 6: Print the resultant array

Step 7: Stop

Output

Enter the size of array 1: 3

enter array1 in sorted order 4 6 8

enter size of array 2: 6

enter array 2 in sorted order 1 3 5 7

merged array is - 1 3 4 5 6 7 8 9 11

PROGRAM NO 2

Aim - Singly linked stack - push, pop, linear search

Algorithm

step 1 : Start

Step 2 : create a newnode with given data when user select a push operation.

Step 3 : check whether the stack is empty or not

If top = = NULL

Then set top = newnode

else :

newnode -> next = top

top = newnode. (end If)

Step 4 : User select pop operation then check whether the stack is empty

else

set temp = top

display. temp - data

Output

1. Push
2. POP
3. Diiplay
4. search
5. Exit

Enter your choice : 1

Enter your element to be Insert : 3

Insertion is succe

1. Push
2. POP
3. Display
4. search
5. Exit

Enter your choice: 4

Enter item to be searched : 34

Item found at location 1

Set top = temp→next

Step 5 : check whether the item is prese

or not

while ptr != NULL

then ptr →data == item

Set flag = 1

else

flag == 0

(item not found)

Step 6 : If the stack is not empty

then

temp → next ! = NULL

temp = temp →next

Step 7 : Display the list

Step 8 : Stop

1. Push

2. POP

3. Display

4. search

5. exit

Enter your choice : 2

The deleted element : 34

1. Push

2. POP

3. Display

4. search

5. Exit

Enter your choice : 3

stack is empty.

PROGRAM NO: 3

Aim: circular queue ADD, delete, search operation using array implementation

Step 1: START

Step 2: check whether the queue is empty

If (front == -1 && rear == -1

Set

queue[rear] = element

else

rear = (rear + 1) % max

queue [rear] = element

Step 3: If the queue is not empty then can be delete element

front = (front + 1) % max

Step 4: Display the elements

If (front == -1 && rear == -1

Set queue is empty

else

repeat while (i <= rear)

seat queue [i]

i = (i + 1) % max

Output

Press 1 Insert an element

Press 2 Delete an element

Press 3 Display the element

Enter your choice

1

enter the element which is to be inserted

10

Press 1 Insert an element

Press 2 Delete an element

Press 3 Display an element

Enter your choice

1

Enter the element which is to be inserted

20

Press 1 Insert an element

Press 2 Delete an element

Press 3 Display an element

Step 5 : Choose the choice and get the result

Step 6 : stop

Enter your choice

1

Enter the element which is to be inserted

30

Press 1 Insert an element

Press 2 Delete an element

Press 3 Display an element

Enter your choice

3

elements in a queue are 10,20,30

Press 1 Insert an element

Press 2 Delete an element

Press 3 Display an element

Enter your choice

2

The dequeued element is 10

PROGRAM NO: 4

Aim : Doubly linked list - Insertion, Deletion searc

Algorithm

Step 1 : START

step 2 : check whether the list is overflow

then print over flow

if (ptr == NULL)

(overflow condition)

else

Read item value

step 3 : check head == NULL

SET ptr -> next = NULL

SET ptr -> prev = NULL

ptr -> data = item

head = ptr

Step 4 : else

SET ptr -> data = item

ptr -> prev = NULL

ptr -> next = head

head -> prev = ptr

head = ptr

then node inserted

step 5 : If the node inserted at the tees

node then

Check the overflow condition

Otherwise

SET

temp = head

Repeat while temp → next = NULL

then

SET

temp = temp → next

temp → next = ptr

ptr → prev = temp

ptr → next = NULL

Step 6 : check the overflow condition

Otherwise inserted at the specified

location

Repeat for i=0; i<loc, and i+

then SET

temp = temp → next

step 7 : Display inserted queue

Step 8 : check head == NULL

The print underflow

step 9 : check head -! next = = NULL

Then SET head = NULL

Deleted node

step 10 : check head → next = = NULL

Then

head = NULL

Otherwise

SET

Ptr = head and check the condition:

IF (Ptr -! next != NULL)

Ptr = Ptr → next

Ptr → prev → next = NULL

Deleted node

Step 11 : check if the queue is empty

Otherwise

Repeat while (ptr ! = NULL)

Check ptr → data = = item

Set flag = 0

Step 12 : Display the final queue

Step 13 : Stop

PROGRAM NO: 6

AIM: Binary search trees - Insertion, Deletion
Search

Algorithm

Step 1 : START

Step 2 : If user Select the insertion operation
then
Create a new BST node and assign
value of it

Step 3 : If root == NULL then
SET temp -> data = data
SET temp -> Left -> right = NULL

Step 4 : If data < node -> data
SET
node -> left and assign the
return value node -> left

Step 5 : data > node -> data
SET node -> right

Step 6 : If the user select the Search
element operation then :

Step 7 : If node = = NULL

Then element not found

Step 8 : If data < node + data

then ~~node~~ → left and assign the return

node → ~~right~~ left

Step 9 : If data > node + data.

then SET

node → right

Step 10 : If the user select the deletion

Step 11 : check node = = NULL

Then element not found

Step 12 : check data < node → data

then SET

node → left

Step 13 : check data > node → data

then SET node → right

check node → right && node → left

Then

// replace with minimum element

in the right subtree

Step 14 : call function del with value

node → right

temp -> data

Otherwise

Set temp = node

Step 15 : If node -> right = = NULL then

Set node = node -> left

free (temp)

Step 16 : If node ! = NULL then

inorder (node -> left)

Display node -> data

Step 17 : Set node -> right

Step 18 : STOP

PROGRAM 7

Aim :- Dis Joint Sets and the associated operations

Algorithm

Step 1 : START

Step 2: Declare the variable of the set

Step 3: Store the user's data and call function make set() then SET i=0

Step 4: Repeat for i<dis n then

SET dis.parent [i] = 1

SET dis rank [i] = 0

SET I = I+1

Step 5: User select union operation then Then read the element to be Perform and store to x set and y

Step 6: perform find operation with x and store result into x set and y set perform step

Step 7 : IF x set == y set then

End of IF]

Step 8 : If rank [x set] < dis rank [y set]

then

SET dis. Parent [x set] = y set

SET dis. rank [x set] = -1

else dis rank [x set] > dis rank [y set]

then

SET dis. Parent [y set] = y set

SET dis. rank [y set] = -1

otherwise

SET dis parent [y set] = x set

SET dis. rank [x set] = dis. rank [x set]

SET dis. rank [y set] = -1

Step 9 : If user choice find operation

then :

If Find x = = Find y then

Display the set

Step 10 : If user select the display

operation then

Set i = 0

Step 11 : Repeat For i < dis.n then

Print dis. parent [i]

SET i = i + 1

step 12 : Repeat For i < dis .n then

Print dis .rank [i]

Set i = i + 1

Step 13 : If dis .parent [x] != x

then

SET dis. parent [x] = find (dis . parent [x])

Step 14 : Display elements

step 15 : Exit