Program: 1 merging

Aim: merge two sorted arrays in a third array.

Algorithm:- MERGING (array1, array2, merge, m, n)

Let array1 and array2 be sorted arrays with m and n elements, respectively. This algorithm merge array1 and array2 into all array 'merge' with m+n elements

```
1. SET i := 0; j := 0; k := 0, [initialise]
2. Repeat while i<m and j<n : [compare]
        If array[i] < array[j] then:
        SET merge[k] = array1[i]
        SET    k = k+1 and i = i+1
else

    SET merge[k] = array2[j]
     SET k = k+1   and j = j+1
[end of If Structure]
[end of loop]
3. Repeate while i<m, then:
          SET merge[k] = array1[i]
           SET i := i+1
SET k = k+1
[End of loop]
4. Repeate while j<n, then:
     SET merge[k] = array2[j].
     SET j = j+1 and k = k+1
  [End of loop]

5. Exit.
```

output

```
Entere the size of the first arraay:
4
Enterthe elements of the first. arraay
1
2
3
4
Enterthe size of the second arraay
5
Entre the elements of the second arraay:
5
6
7
8
9
the merged arrays
1
2
3
4
5
6
7
8
9
```

program: 2 SINGLY LINKED STACK

Aim: Singly linked stack - push, pop, Linear search

Algorithm :-

```
1. Start
2. select push operation, then
3. create a new node with the given data
4. If top = = NULL then:
     (check whether the stack is empty)
5 SET top = new node
  SET newnode -> next = NULL
  ELSE:
     SET newnode -> next = top
     top = new node
  [end of if structure]
6. If user select pop operation then
7 If top = = NULL then:
     (check whether stack is empty)
  display "stack is empty"
Else:
  SET temp = top
(create a temporary node & set it to top)
```

```
display tem ->data
8. SET top = temp -> next
     (make Top point to the next node)
9. Free (temp)
     (Delete the temporary node)
10. If user select search operation then
11. Declare a pointer variable temp and the
     variable key that holds the value to be searched
12. SET temp = Top
     SET flag = 0
13. Repeate while temp != NULL
          If temp -> data = = key then:
          display "element found at location"
       SET flag = 1
       [End of If Architecture]
       ~~Goto~~ go to step 14
Else:
       SET temp = temp -> next
    [End of while]
14. If flag = = 0  then:
```

```
display "element not found"
 [end of If structure]

15 : If user select display operation then
16 : declare a pointer node ptr
        SET node ptr = top

17 : If node ptr = = NULL then:
        display "stack is empty"
        [end of If structure]
18 : while node ptr != NULL then:
      print node ptr -> data
      SET node ptr = node ptr -> next
If node ptr != NULL then:
         print "-->"
   [End of If structure]
  [End of while]
19 : Exit.
```

output

```
1. push
2. pop
3. display
4. Search
5. exit
Enterthe choice:1
Enter the element tobe inserted:2
insertion is successfull

1. push
2. pop
3. display
4. Search
5. Exit
Enterthe choice:1
Enterthe element tobe inserted:4
insertion is successfull
```

```
1. push
2. pop
3. display
4. Search
5. Exit
Enter the choice:1
Enter the element tobe inserted:10
Insertion is success fcell

1. push
2. pop
3. display
4. Search
5. Exit
Enter the choice:2
element Deleted:10

1. push
2. pop
3. Display
4. Search
5. Exit
```

Enter the choice:3

4->2->null

1.push
2.pop
3.display
4.search
5-exit

Enter the choice:4

Enter the item which is to be searched:2

Item found at location:2

~~menu~~

1.push
2.pop
3.Display
4.search
5.Exit

Enter the choice:5

program: 3 circular queue

AIM: circular queue operations

```
Algorithm:- 1. start
        2. If user selects the insertion operation
           then
3. declare a variable item with given value
4. If front == 0 && rear == size-1
   If front == rear+1 then
   Display "queue overflow"
   [End of if structure]
5. If front == -1 then
      SET front = 0
      SET rear = 0
   [End of If structure]
6. If rear == size-1
   SET rear = 0
   Else:
      SET rear = rear + 1
      [End of If structure]
```

```
7. q[SET (q[rear] = Item
8. If user select deletion operation then
9. If front = = -1 then
      display a "queue underflow
      [End of If Structure]
10. If front = = rear then
      SET front = -1
      SET rear = -1
      [End of If Structure]
11. If front = = size -1 then:
      SET front = 0
      Else
      SET front = front +1
      (End of If Structure]
12. If user select the display operation then
13. SET front-pos = front
      SET rear_pos = rear
14. If front = = -1 then:
      Display "Queue is empty"
      [End of If Structure]
```

```
15: If front_pos <= rear_pos then:
    Repeat while front_pos <= rear_pos then:
        print cq[front_pos]
        SET front_pos = front_pos+1
    End of while

  else:
    Repeate while front_pos <= size-1
        print cq[front_pos]
        SET front_pos = front_pos+1
    [End of while]
16: SET front_pos = 0
17. Repeate while front_pos <= rear_pos then:
        display cq[front_pos]
        set front_pos = front_pos+1
    [End of while]
    [End of IF]
18: If user select search operation then
19: Declare a variable set with value be
    Searched
20: declare a temporary variable temp then
```

```
    SET temp=sea

21: SET i=front

22: Repeate for i <= rear then:
     If te== cq[i] then:
    Print .j+1
    SET j=j+1
    [End of If]
     If j==0 then
       Display "Item_not found"
    [End of If structure]
     SET i=i+1
     [End of for loop]

23: Exit
```

output

```
menu
1. insert
2. Delete
3. Display
4. Search
5. exit
enter the choice:1
Enter the number to insert:10
menu
1. insert
2. Delete
3. Display
4. Search
5. exit
enter the choice:1
Enter the number to insert:20
menu
1. insert
2. Delete
3. Display
4. search
5. exit
Ente
```

```
Enter the choice: 1
Enter the number to insert: 30
menu
1- insert
2- Delete
3- Display
4- Search
5- exit
Enter the choice: 3
   10, 20, 30
menu
1- insert
2- Delete
3- Display
4- Search
5- exit
Enter the choice: 4
Enter the element which is to be searched: 30
Item found at location 3
```

```
menu
1. Inseat
2. Delete
3. Disxlay
4. Search
5. exit
Enterthe choice:2
 10 was delead !
menu
1. Inseat
2. Delete
3. Displey
4. Search
5. exit
select the choice:5
```

program: 4

AIM: set operations

```
Algorithm:- 1. start
2. If user select the union operation then:
3. declare two array set1[i]
   and set 2[i], Declare two variable n1,n2
   for holding the size of two arrays
4: read elements into the arrays
      set1[i] and set2[i]
5: If $n_1 == n_2$ then:
6: SET i=0
7. repeate for i<n2 then
8: SET set 3[i] = set1[i] || set2[i]
8: SET i=i+1
   [end of for loop]
9: SET i=0
10. Repeat for i<2 then
    print set3[i]
11. SET i=i+1
```

```
{End of for loop}
[End of IF}
ELSE :
    print "size are not equal"
    Exit
12: If user select insertion operation then
13: declare two array set1[i] and set2[i]
    with size n1, n2 respectively and read elements
    to the arrays
14. If n1 = = n2 then
15. SET i = 0
16: repeate for i<n2 then
17. SET set3[i] = set1[i] & set2[i]
18: SET i = i+1
    {End of for loop}
19: SET i = 0
20: repeate for i<n2 then:
            print set3[i]
21: set i = i+1
    (End of for loop)
    {End of IF}
    ELSE
    print "size are not equal"
    Exit
22. If user select the substraction then
23: declare two array set1[i] and set2[i]
    with n1,n2 size respectively and input the
    element to the array
24: If n1 = = n2 then
25: set i = 0
26: Repeat for i<n2 then:
        set set3[i] = set1[i] ee!. set2[i]
27: i = i+1
        [end of for loop]
28: SET i = 0
29: Repeat for i<n2 then print set3[i]
~~29: Repeat for i<n2 l~~
30: Set i = i+1
        [end of for loop]
        [end of if]
Else
    "print size are not equal"
31: Exit
```

output

```
press1 for union
press2 for intersection
press3 for subtraction
press 4 for exet
enter choice1
enter the size of set1
3
enter the elements of set 1
1
2
3
enter the size of set2
3
enter the elements of set2
1
2
3
union: 123
press1 for union
press 2 for intersection
press 3 for subtraction
press 4 for exet
enter the choice: 2
```

```
Entre the size of set1
3
Entre the elements of set1
1
2
3
Entre the size of set2
3
Entre the elements of set2
3
4
5
Intersection:3
press1 for union
press2 for intersection
presss3 for subtraction
press4 for exit
Entre your choice:3
Entre the size of set1
3
Enter the elements of set1
1
2
3
Entre the size of set2
3
Entre the elements of set2
4
3
2
differnce:1
```

```
press 1 for union
pass 2 for intersection
press3 for subtraction
press 4 for exit
enter the choice :4
```

program: 5 Binary search Tree :

AIM: Binary Search tree operations

r

Algorithm:- 1. start

2. If user select the insertion operation then

3. create a new BST node and assign value to it

4. create tree (node, data) // all the create tree function with the root value and the data entered by user

5. If root = = NULL then;

6: Declare a temporary variable is temp

SGT temp -> data = data

SGT temp -> left -> right = NULL;

return the new node temp to the calling function [end of If]

7. If data < (node -> data)

8. call the create node function with node -> left and assign the return value in - node -> left

9. node -> lef = create tree (node -> lef, data)

[end of If structure]

9. If data > node -> data

10. call the create tree function with node -> right and assign the return value in node -> right
node -> right = create tree (node -> right data)
[end of If structure]

11. return the original root pointer node to the calling function.

12. If the user select the search element operation then.

13. search (node, data) // call the search function with root value and the element the to be searched.

14. If node = = NULL
print "element not found"
[end of If]

15. If data < node -> data then : call the search function with node -> left and assign the return value in node -> left
node -> lef = search (node -> left, data
[end of If structure]'

```
16: If data > node -> data then
      call search function with node -> right
      and assign the return value in node -> right
    node -> right = search (node -> right, data
      [End of If Structure]
    Else:
      print "Element found %s" node -> data
17. return the original root pointer 'node'
      to the calling function.
18. If the user select the deletion operation then
19: del (node, data) // call the del function with
      root value and the element to be deleted
i:  declare a temporary variable temp
19  If node == NULL then:
          print "Element not found"
      [End of If]
20: If data < node -> data then
      call the del function with node -> left and
      assign the return value to node -> left
    node -> left = del (node -> left. data)
    [End of If]
```

```
21 : If data > node -> data then call the del function
    with node -> right and assign the return
    value n
    node -> right
    [End of If]
22 : ELSE
    // delete this node and replace with other
    minimum element in the right sub tree or
    maximum element in the left sub tree.
23 : If node -> right && node -> left // replace with
    minimum element in the right sub tree.
24 . ~~SET~~ call find min function with node ->
    right then Return value assign in temp
                  Goto step 30
    SET temp = function (node -> right)
    ~~SET temp = findmin~~
    SET node -> data = temp -> data
    // replaced it with some other node
25 . call function del with value node -> right
        temp -> data and return value assign
        in node -> right
    ELSE:
26 : SET temp = node
```

//If there is only one or zero children then we can directly remove it from the free and connect its parent its child.

```
27: If node->left = = NULL then:
      SET node = node -> right
    Else
28: If node->right = = NULL then:
      SET node = node->left.
29: free (temp)
      [End of If]
      [End of If]
      [End of If]
30: find min (node)
31: If node = = NULL then
          return NULL
          Goto Step 24
      [End of If]
32: If node -> left then call the function find min
    with value (node -> left) then return the value
    to calling function. (node -> left)
    Else
    return node
    Goto step 24
    [End of If
33 If the user select the Display option then
34 Inorder (ode
    call the inorder function with root value
35: nod != NULL then
    inorder (node -> left)
    call the function inorder with value node->left
36 print node -> data
    inorder (node -> right
    call the function inorder with value node->right
    [End of If]
37. Exit
```

output

```
1. Insert in Binary tree
2. Delete from Binary tree
3. Inorder traversal of Binary tree
4. search
5. Exit
Enter the choice: 1
 Enter new element: 20
   root is 20
 Inorder traversal of binary tree: 20
1. insert in Binary tree
2. Delete from Binary tree
3. Inorder traversal of Binary tree
4. Search
5. exit
Enter the choice: 1
Enter new element: 25
Inorder traversal of Binary tree: 20 25
```

```
1. Insert in Binary tree
2. Delete from Binary tree
3. Inorder traversal of Binary tree.
4. Search
5. exit
 Enter the choice:4
  Enter the element to be searched:25
 Element 25 which was searched is found
1. insert in Binary tree
2. Delete from Binary tree
3. inorder traversal of Binary tree
4. search
5. exit
 enter the choice:5
```

programs Double Linked List

AIM: Doubly linked list operation.

```
Algorithm: 1. Start
2. If user select the Insert operation at beginning
   then
3. If head = = NULL then:
4. perform step S6 to S9 (call the function create)
5. SET Head = temp
   temp1 = head
6. Else
7. perform step S6 to S9 (all the function create)
8. SET temp -> next = head
   SET head -> prev = temp
   SET head = temp
   [End of If structure]
9. If user choose the operation insert node at end
   then.
10 else If head = = NULL then;
11. perform S6 to S9
12 SET head = temp, SET temp1 = head
```

```
13 •Else
14• perform step 56 to 59
15• SET temp1->next=temp
    SET temp->prev=temp1
    SET temp1 = temp
    [End of If Structure]
16 •If user choose insert at any position then
17• Read the position and Store It in to the Variable
    pos
18• SET temp2=head
19• If pos<1 || pos>=count+1 then
    Display "position out of range to insert"
    Exit
    [End of If Structure]
20• If head==NULL && pos!=1 then
    Display "Empty list cannot insert other than
    1st position"
    Exit [End of If Structure]
21: If head == Null && pos==1 then
22: perform step 56 to 59 (call function create)
```

```
23: SET head = temp
    SET temp1 = head
    [End of if]
    Exit
24. Else
    Repeate
25. while i< pos then:
26  SET temp2 = temp2 -> next
    SET i = i+1  [End of while]
27: peeform step 56 to 59
28: SET temp -> prev = temp2 2
    SET temp -> next = temp2 => next'
    SET temp2 -> next = temp
29: If user choose the operation detection then:
30: Read the position the If store into the variable pos
31: SET Temp2 = head
32: If pos <1||pos> = count+1 then Display
    "position out of range to delete'
    [End of If structure]
    exit.
```

```
33. IF head == NULL then
      Display "Empty list no elements to delete"
      [End of IF structure]
      Exit
34. ELSE
35. Repeate while i<pos
      SET temp2 = temp2 ->next
      SET i = i+1
      [End of while]
36. IF i==1 then
      IF temp2->next == NULL then
     Display "node deleted from list"
37. free temp2
     SET temp2 = head = null
          [End of IF structure]
      Exit
36. IF temp 2-> next = NULL then:
     SET temp2-> prev->next = NULL.
    free temp2
    Display "node deleted from list"
      [End of IF structure]
     Exit.
    free temp2
37. SET temp2 => head = NULL
    [End of If Structure]
  Exit

38  If temp2 -> next = NULL then:
    SET temp2 -> prev -> next = NULL
    free temp2
   Display "node deleted from list"
    [End of If Structure]
    Exit

39: SET temp2 -> next -> prev = temp2 -> prev
38 :If i=1 then
     temp2 -> prev -> next = temp2 -> next
     [End of If Structure]
39. If i = =1 then
   set head = temp -> next
   Display "node deleted"
40: free temp2
    [End of If Structure]
41: SET count = count - 1
42: If user select display operation
```

```
then 43: SET temp2 = head
43: SET temp 2 = = NULL
   Display "List empty todisplay"
    [End of IF Structure]
    Exit
45: while temp2 -> next != null then
     print temp2 -> n
       SET temp2 = temp2 -> next
       [End of while]
46: print temp2 -> n
47: IF user selects search preform then.
48: SET temp = head
49: IF temp2 = = NULL then
     Display "List Empty to search for data"
       [end of IF Structure]
       Exit
50: Read the value to be search and store it in
    to the variable data
51 while temp 2 != NULL then
52 IF temp2 -> n = = data then
```

```
53: Else
    SET temp2 = temp2 -> next
    SET count = count +1
    [End of If]
    [End of while]
54: Display "not found"
55: Exit
56: //when call create function
 .  SET temp -> prev = NULL //create()
57: SET temp -> next = NULL
    Display "Entere value to node
58: read data and assign it into
       temp->n
     temp->n = data
59: count = count +1
60: Exit
```

output

```
1. insert at begening
2. insert at end
3. insert at position
4. Detete
5. Display
6. search
7. Exit
Enter choice:1
.Enter the value of node 5
1. insert at beginning
2. insert at end
3. insert at position
4. Delete
5. Display
6. search
7. exit
Enter choice:1
Enter the value of node:10
```

```
1. Insert at beginning
2. Insert at end
3. Insert at possition
4. Delete
5. Display
6. Search
7. exit
Enter choice: 2
Enter the value to node: 2
1. Insert at beginning
2. Insert at End
3. Insert at position
4. delete
5. Display
6. Search
7. Exit
Enter the choice: 3
Enter the position to be inserted: 2
Enter the value to node: 13
```

```
1. insert at beginning
2. insert at end
3. insert at position
4. Delete
5. Display
6. Search
7. exit
entre the choice:4
entre the position tobe deleted:2
node deteed
1. insert at beginning
2. insert at end
3. insert at position
4. Delet
5. Display
6. search
7. exit
entre choice:6
entre the value to search: 10
element found in 1 position
```

```
1. insert at begenning
2. insert at end
3. insert at position
4. Delete
5. Display
6. Search
7. exit
  enter the choice:7
```

program Disjoint

AIM : Disjoint Set operations

program:

```
1. Start
2. Read the number of element from user and store
   it in to the dis.n
3. call function makeset() then
4. SET i=0
5. Repeat for i<dis.n then
     SET dis.parent[i]=i
     SET dis.rank[i]=0
     SET i=i+1
     [end of for]
6. user select the union operation then
7. Read the element to perform union and store
   in to x & y respectively
8.  // perform and operation with x, store result
```

```
in toxset naeform step
9. If X set == Y set then
    [End of If]
10. If is ranlh [xset] < dis.ranlh [yset] then:
    SET dis.parent [xset] = yset
    SET dis ranlh [xset] = -1
    [End of If]
11. else if dis.ranlh [xset] > dis.ranlh
    [yset] then
    SET dis.parent [yset] = x set;
    SET dis ranlh [yset] = -1
    [End of If]
12: else
13: SET dis.parent (yset) = xset
    SET dis.ranlh [xset] = dis.ranlh[xset]+1
    SET dis.ranlh [yset] = -1
14. If user choose find operation then
15: Read the element to check if and store the
    value into the variables x and y respectively
```

```
16: If find x == find y then
    display " connected components"
17: else
    display " not connected components"
18: If user select the display operation then
19: SET i=0
20: repeate for i < dis.n then
    print dis.parent[i]
     SET i=i+1
   [end of for loop]
21: SET i=0
22: Repeat for i < dis.n then:
    print dis.rank[i]
     SET i=i+1
    [end of for loop]
23: If dis.parent[x] != x then
    SET:
        SET dis.parent[x]= find (dis.parent[x])
        return dis.parent[x]
24: Exit
```

output

```
How many elements? 4
menu
1. insertion
2. search
3. display
enter the choice
1
enter element to perform insertion
3
4
Do you wish to continue(1/0)
1
menu
1. insertion
2. search
3. display
enter choice
1
enter the element to perform insertion
5
6
```

```
Do you want to continue(1/0)
1
menu
1. insertion
2. find
3. display
Enter choice:
3
parent array
3 1 2 3
Rank array
-1 0 0 1
Do you want to continue(1/0)
0
```