Program No : 1

Merge Two Sorted Arrays and Stored in a Third Array

Step 1 : Start
Step 2 : Declare the variable
Step 3 : Read the size of the first array
Step 4. : Read elements of first array in
-Stored order
Step 5 : Read the size of the second array.
Step 6. : Read 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[j] then c[k++] = b[j++]
Step 9 : else c[k++] = a[i++]
Step 10 : Repeat step 11 while i<m.
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 merged array
Step 17 : End.

Output

size of first array

2

Enter value in sorted order

1

3

size of second array

3

Enter value in sorted order

2

4

5

Merged array

1

2

3

4

5

Program No: 2

Singly Linked Stack - Push, Pop, linear Search.

Step1 :- Start

Step2 :- Declare the node and the required variable.

Step3 :- Declare the functions for push, pop, display and search an element

Step4 :- Read the element 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 variable values 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 top = newNode & then print insertion is successful.

Step 6 :- If user choose to pop 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 pointed variable temp and initialize it to top.

Step 6.3 & Print the element that being deleted.

Step 6.4 & set temp = tempt → next

Step 6.5 & free the temp.

Step 7 & if the user choose the display then call the function to display the element

Step 7.1 & check if top == Null then print stack is empty.

Step 7.2 & else declare a pointer variable temp & initialize 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 pointed variable ptr and other neccessary 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 Step 8.6 to 8.8 while Ptr != null.

Step 8.6 :- check if ptr → data == item. then point element founded and to be located and set flag = 1

Step 8.7 :- else set flag = 0

Step 8.8 :- increment the by one and set Ptr = Ptr → next.

Step 8.9 :- check if flag = 0 then print the element not found

Step 9 :- End.

Output

Menu

1. Push

2. Pop

3. Display

4. Search

5. Exit

Enter your choice : 1

Enter the value to be insert : 2

Insertion is successful.

Menu

1. Push

2. pop

3. Display

4. Search

5. Exit

Enter your choice : 1

Enter the value to be inserted : 34

Insertion is Successful.

Menu

1. Push

2 Pop

3. Display

4. Search
5. Exit
Enter your choice: 1
Enter the value to be inserted: 45
Insertion is successful.
Menu:
1. push
2. Pop
3. display
4. search
5. Exit
Enter your choice: 1
Enter the value to be inserted: 57
Insertion is successful.
Menu
1. Push
2. Pop
3. display
4. search
5. Exit
Enter your choice: 3.
57 → 45 → 34 → 2 → Null.
Menu.
1. Push
2 Pop

3. Display
4. Search
5 Exit
Enter your choice : 2.
Deleted element : 57.
Menu
1. Push
2. Pop
3. Display
4. Search
5. Exit
Enter your choice : 4
Enter the item you want to search : 2
Item found at location 3.
Menu
1. push
2 pop
3 Display
4. Search
5. Exit
Eter your choice : 5
Exit.

Program No: 3

Circular Queue -

Add, Delete, Search

Step 1 :- Start

Step 2 :- Declare queue and required variable.

Step 3 :- Declare the function for enqueue, dequeue, display and search.

Step 4 :- Read the choice from the user to enqueue, dequeue, display and search

Step 5 :- If the user choose the option enqueue then read the element to be inserted from the user, then call the function enqu and pass the values to the function.

Step 5.1 :- check if front == -1 and rear == -1 then set front = 0, rear = 0 and set queue[rear] = element

Step 5.2 :- else if rear+1 mod max == front or front == rear+1 then print queue is overflow

Step 5.3 :- else set rear = rear+1 mod max, and set queue[rear] = element

Step 6 :- If the user choose the option dequeue

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 to be deleted. Then set front = -1 and rear = -1

Step 6.3 :- else print the element to be dequeue set front = front + 1 mod max.

Step 7 :- if the user choose the option to display the queue then 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 mod max.

Step 8 :- if the user choose to search an element in the queue then call the function to search an element in 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 i by 1

Date
Page

Step 8 :- check if i==0 then print item not found.

step 9. :- End.

Output

Menu

1. Insert

2. Delete

3 display

4. search

5 Exit

Enter your choice : 1

Enter the value to be inserted : 4.

Menu

1. Insert

2 Delete

3. Display

4. Search

5. Exit

Enter your choice : 1

Enter the value to be inserted : 8

Menu

1. Insert

2. Delete

3. Display

4. Search

5 Exit

Enter your choice : 1
Enter a number to be inserted : 2
Menu:
1. Insert
2. Delete
3. Display
4. Search
5 Exit
Enter your choice : 1
Enter the value to be inserted : 10
Menu
1. Insert
2. Delete.
3 Display
4. Search
5 Exit
Enter your choice : 1
Enter the value to be inserted : 13
Menu.
1. Insert
2. Delete.
3 Display
4. Search
5 Exit
Enter your choice : 3

4 8 2 10 13

Menu

1. Insert

2. Delete

3. Display

4. search

5. Exit

Enter your choice : 2.

4 was deleted.

Menu.

1. Insert

2. Delete.

3 Display

4. Search

5. Exit

Enter your choice : 4.

Enter the element which is to be searched : 2.

Item found at location 3.

Program No: 4

Doubly Linked List - Insertion, Deletion, Search

Step 1 :- Start

Step 2 :- Declare a Structure and related Structure variable

Step 3 :- Declare function to create a node. insert a node in the beginning. insertion at the end. insertion at the given position display the list and search an element in the list

Step 4 :- Define a function to create a node, declare the required variable.

Step 4.1 :- Set memory allocated to the node=temp then set temp→ Ptr = 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 increament count by 1

Step 5 :- Read the choice from the user to perform different operations on the list

Step 6 :- If the user choose to perform insertion

operation at the beginning then call the function to perform the insertion.

Step 6.1 :- check if head == null then call the function to create a node, perform step 4 to step 4.3

Step 6.2 :- set head = temp and temp1 = head

Step 6.3 :- else call the function to be create a node. perform step 4 to step 4.3

Then set temp → next = head set head → Prev = temp and head = temp.

Step 7. :- if the used choose to perform insertion operation 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 newnode then set temp = head and then set head = temp1

Step 7.2 :- else, call the function to create a newnode then set temp1 → next = temp, temp → prev = temp1 and temp1 = temp

Step 8 :- If the user choose to perform insertion operation in the list at any position then call the function to perform the insertion operation.

Step 8.1 :- Declare the neccessary variable.
Step 8.2 :- Read the position where the node need to be inserted. set temp2 = head
Step 8.3 :- check if pos<1 or pos>count+1 then print the position is out of range.
Step 8.4 :- check if head == null and pos = 1 then print "Empty list cannot insert other then 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 = temp1
Step 8.6 :- while i<pos then set temp2 = temp2 → next then increment i by 1
Step 8.7 :- call the function to create a new node and then increment set temp → prev = temp2 temp → next = temp2 → next, temp2 → next = temp
Step 9 :- If the user choose to perform deletion operation in the list then call the function to perform the deletion operation
Step 9.1 :- Declare the mumary variable.
Step 9.2 :- Read the position where node need 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 temp2 = temp2→next and increment i by1

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 temp2→next == null then temp2→prev →next = null then free(temp2) then print node deleted.

Step 9.8 :- temp2→next→prev = temp2→prev. then check if i != 1 then temp2→prev→next = temp2→next

Step 9.9 :- check if i=1 then head = temp2→next then print node deleted then free temp2 and decrement count by1

Step 10 :- if the user choose to perform the display operation then call the function to display the list

Step 10.1 :- set temp2 = h.

Step 10.2 :- check if temp2 = null then print list is empty.

Step 10.3 :- while temp2→next != null then print temp2→n then temp2 = temp2→next

Step11 :- If the user choose to perform the search operation then call the function to perform search operation

Step11.1 :- Declare the neccessary variable.

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.

Step11.5 :- while temp2 != null then check if temp2 -> n == data then print element found at position count+1

Step11.6 :- else set temp2 = temp2 -> next and increment count by 1

Step 11.7 :- print element not found in the list

Output

1. insert at beginning
2 insert at end
3 insert at position i
4 delete at i
5 display from beginning
6 search for element
7 Exit

Enter your choice: 1
Enter value to node : 30

Enter your choice : 1
Enter value to node : 25

Enter your choice : 1
Enter value to node : 15

Enter your choice : 1
Enter value to node : 10

Enter your choice : 1
Enter value to node : 5
Enter your choice : 5

linked list element from beginning :
5 10 15 25 30

Enter your choice : 2
Enter value to node : 20

Enter your choice : 5
Linked list from beginning :
5 10 15 25 30 20

Enter your choice : 4
Enter position to be deleted : 6
Node deleted from list

Enter your choice : 5
Linked list element from beginning :
5 10 15 25 30

Enter your choice : 3
enter position to be inserted : 4
Enter value to the node : 18

Enter your choice : 5
Linked list element from beginning :
5 10 15 18 25 30

Enter your choice : 6
Enter value to search : 15
Data found in 3 position.

Enter your choice : 4.
Enter position to be deleted : 5
Node deleted

Enter your choice : 5
Linked list element from beginning:
5 10 15 18 30.

Enter your choice : 7.
Exit.

Program No: 5

Set data Structure and Set Operations using Bit String

Step 1 :- Start

Step 2 :- Declare the neccessary variable

Step 3 :- Read the choice from the user to perform set operations

Step 4 :- If the user choose to perform union

Step 4.1 :- Read the cardinality of two sets

Step 4.2 :- Check if m!=n then print element cannot perform union.

Step 4.3 :- else read the elements in both the sets.

Step 4.4 :- Repeate the step 4.5 to 4.7 until i<m

Step 4.5 :- d[i] = A[i] | B[i]

Step 4.6 :- Print d[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 two 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 :- Repeate the step 5.3 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 :- print if the user choice to perform set difference operations

step 6.1 :- Read the cardinality of two sets

step 6.2 :- check if m1 = 0 then print cannot perform set difference operation

step 6.3 :- else read the elements in both sets

step 6.4 :- Repeate 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 C[i] = 1

step 6.8 :- increment i by 1

step 7 :- Repeat the step 7.1 to 7.2 until i<m.

step 7.1 :- Print C[i]

step 7.2 :- Increment i by 1

Output

input choice to be perform
1 union 2. intersection 3 difference 4 Exit
choice 1
Enter cardinality of first set : 4
Enter cardinality of second set : 4

Enter elements of first set : (0/1) =
1
0
1
0
Enter elements of second set : (0/1)
0
1
0
0
Elements of set1 union set2 : 1110
Input choice to perform :
1. union 2. intersection 3. difference 4 Ex
choice 2.
Enter cardinality of first set : 4
Enter cardinality of second set : 4
Enter elements of first set : (0/1)

1
1
0
0
Enter elements of second set: (0/1)
1
0
0
1
Elements of set 1 intersection set 2 : 1000
Input choice to perform
1. Union 2. Intersection 3. Difference 4. Exit
Enter cardinality of first set : 4
Enter cardinality of second set : 4.
Enter elements of first set : (0/1)
1
1
0
0
Enter elements of second set : (0/1)
0
1
0
1
Elements of set1 - set2 & 1000
program exit successfully.

Program No: 6

Binary Search Tree-
Insertion, Deletion, Search

Step1 :- Start

Step2 :- Declare a structure and structure pointers 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 choose to perform insertion operation then read the value which is to be inserted to the tree from the user

Step5.1 :- Pass the value to the insertion pointer and also the root pointer

Step5.2 :- Check if !root then allocate memory for the root

Step5.3 :- Set the value to the info part of the root and the set left and

right part of the root to null and return root

Step 5 :- 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 the 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.

Step 6.1 :- check if not ptr then print node node 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 delete 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 6.5 :- else if ptr → left == null then set P1 = Ptr → right and free ptr, return P1

Step 6.6 :- else if ptr → right == null then set P1 = Ptr → right and free ptr, return P1

Step 6.7 :- else set P1 = Ptr → right and P2 = Ptr → right

Step 6.8 :- while P1 → left not equal to null,
set P1 → left = Ptr → left and free Ptr,
return P2.

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 return root.

Step 8 :- If the user choose to perform

traversal then call the traversal function and pass the root pointer.

Step 1 : if root not equal to null recursively call the function by passing root→left

Step 2 : print root→info.

Step 3 : call the traversal function recursively by passing root → right

Output

1 insertion in binary tree
2 Delete from binary tree
3 inorder traversal of binary tree
4 Search
5 Exit
Enter your choice: 1
Enter new element: 15
root is: 15
inorder traversal of tree is: 15
Insertion in binary tree.
2. Delete from binary tree
3 inorder traversal of binary tree
4 search.
5 Exit
Enter your choice: 1
Enter new element: 30
root is: 15
inorder traversal of binary tree is:
1. insert in binary tree.
2 Delete from binary tree
3. inorder traversal of binary tree
4 search
5 Exit

Enter your choice: 1
Enter root element: 100
root is 15
Inorder traversal of binary tree is: 15 30 100
1. Insert in binary tree
2. Delete from binary tree
3. Inorder traversal of binary tree
4. Search
5. Exit
Enter your choice: 1
Enter new element: 50
root is 15
Inorder traversal of binary tree is
15 30 50 100
1. Insert in binary tree
2. Delete from binary tree
3. Inorder traversal of binary tree
4. Search
5. Exit
Enter your choice: 4.
Search operation in binary tree
Enter the element to be searched: 15
Enter 50 which was searched is found
and it = 15

1. Insertion in binary tree
2 Delete from binary tree.
3 Inorder traversal of binary tree
4. Search
5. Exit
Enter your choice : 2.
Enter the element to be deleted : 20.
1. Insert in binary tree.
2. Delet from binary tree
3. Inorder traversal of binary tree
4. Search.
5. Exit
Enter your choice : 3.
Inorder traversal of binary tree is :
15 50 100.
1. Insert in binary tree.
2. Delete from binary tree
3. Inorder traversal of binary tree
4. Search.
5. Exit.
Enter your choice : 5

End of program

Program No:7.

Disjoint Sets 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 : Repeate 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 : Repeate 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 : Repeate 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
1 to the function
step 5.1 : check if dis parent [a] != a then set
the return value to dis parent [x]
step 5.2 : return dis parent [x]
step 6 : Declare a function union and pa

find variable of root y:

Step 6.1 : set z set to find(x)

Step 6.2 : set y set to find(y)

Step 6.3 : Check if x set = = y set then return

Step 6.4 : check if dis rank [xset] < dis rank [yset] then

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 ~~parent~~ rank [y set]

Step 7 : Read the number 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

step11 : if the user choose to perform find operation read the element to check if connected

step11.1 : check if find(x) == find(y) then print connected component

step11.2 : else print Not connected

Step 12 : if the user choose to perform display operations call the function display set

Step 13 : End.

Output

How many elements? 5

Menu

1. Union

2. Find

3. Display

Enter choice 1

Enter elements to be perform 1

2

Do you wish to continue (1/0) 1

Menu

1. union

2 Find

3 Display.

Enter choice 3

Parent Array

0 1 1 3 4

Rank Array

01-100

Do you wish to continue (1/0) 1

Menu

1. union

2. Find

2. Display

entered choice 2
Enter element to check if connected
component 2.

4
connected component.
Do you wish to continue (Yes) 0.

Exit.