program no:-1

Aim:- merge two sorted arrays and store in a third array.

Algorithm

Step-1:- Start

Step-2:- Declare the variables

Step-3:- Read the size of first array

Step-4:- Read elements of first array in sorted order

Step-5:- Read the size of second array

Step-6:- Read the elements of second array in sorted order

Step-7:- Repeat Step 8 and 9 while i<m and 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 steps 11 while i<m

Step-11:- c[k++] = a [i++]

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:-

enter number of elements in array1:-4

enter array 1 in sorted order : 1 3 5 7

enter number of elements in array2: 4

enter array 2 in sorted order:- 1 4 6 7

First array is : 1 3 5 7

second array is : 2 4 6 7

merged array is : 1 2 3 4 5 6 7 7

Scanned with Fast Scan

program no:-2.

Aim:- stack operations

Algorithm

Step-1:- Start

Step-2:- Declare the node and the required variables

Step-3:- Declare the functions for push, pop, display and search an element.

Step-4:- Read the choice from the user.

Step-5:- 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 value 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 newNode → ~~next~~ next = top

Step-5.5:- set top = newNode ~~of~~ and 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 pointer variable temp and initialize it to top.

Step-6.3:- print the element that being deleted.

Step-6.4:- set temp = temp → next.

Step-6.5:- Free the temp

Step-7:- if the user Choose the display then call the function to display the element in the stack.

Step-7.1:- check if top==null then print stack is empty

Step-7.2:- else declare a pointer variable temp and 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 pointer variable ptr and other necessary 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 tr! = null

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

Step-8.7:- else set flag = 0

Step-8.8:- Increment i by 1 and set ptr = ptr → next

Step-8.9:- Check if flag = 0 then point the element not found.

Step-9:- End.

output:-

choices

1. push

2. pop

3. Display

4. search

5. exit

enter your choice :1

enter the value to be inserted:7

Insertion is successfull

Choices:

1. push

2. pop

3 Display

4. search

5. exit

enter your choice : 2

popped element:14

choices

1. push

Scanned with Fast Scan

2. push

3. pop

4. Display

5. exit

enter your choice: 3

7 Null

Choices

1. push

2. pop

3. Display

4. Search

5. exit

enter your choice: 4

enter the item to be searched: 2

item not found

Choices

1. push

2. pop

3. Display

4. search

5. exit

enter your choice: 5

Program no:- 3

Aim:- Circular Queue - Add, Delete, Search.

Algorithm

Step-1:- Start

Step-2:- Declare a queue and other variables.

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

Step-4:- Read the choice from the user.

Step-5:- if the user choose the choice enqueue. then read the element to be inserted from the user and call the enqueue function by passing the value.

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 % max == front or front = rear+1 then point queue is overflow.

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

Step-6:- if the user choice is 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 is to be deleted. Then set front = -1 and rear = -1.

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

Step-7:- if the user Choice is 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 % max.

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

Step-8.3:- Check if c==0 then print item not found.

Step-9:- End

Scanned with Fast Scan

output:-

choices:-

1. Insert

2. Delete

3. Display

4. Search

5. Excit

Enter your choice:-

1

Enter the number to be inserted

7

choices:-

1. Insert

2. Delete

3. Display

4. Search

5. Excit

Enter your choice:-

1

Enter the number to be inserted:

14

Scanned with Fast Scan

choices:

1. Insert

2. Delete

3. Display

4. Search

5. Exit

enter your choice :
2

7 was deleted

choices:-

1. Insert

2. Delete

3. Display

4. Search

5. Exit

enter your choice:
3

enter the element to be inserted:-
14

choices:-

1. Insert

2. Delete

3. Display

4. search

5. Exit

enter your choice:

4

enter the element to be search:

14

item found at location:-

0.

Scanned with Fast Scan

program no:- 4

Aim :- Doubly linked list - Insertion, Deletion, 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 begining 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 required 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

Step-5:- Read the Choice from the user to perform different operation on the list.

Scanned with Fast Scan

Step-6:- if the user choose to perform insertion operation at the begining 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 4.3.

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

Step-6.3:- 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 to 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 temp → next = temp. temp → prev = temp1 and temp1 = temp.

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

Step-8.1:- Declare the necessary variable.

Step-8.2:- Read the position where the node need to the 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 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 = 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 set temp→prev=temp2.temp →next =temp2 →next→prev= temp.temp2→ next= temp.

Step-9:- if the user choose to perform deletion operation is the list then all the function to perform the deletion operation.

Step-9.1:- Declare the necessary variables.

Step-9.2:- Read the position where node need to be deleted Set temp2 = head.

Step-9.3:- Click 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 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 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 by 1.

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

Step-10.1:- set temp2 = n

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

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

Step-11:- if the user choose to perform the Search operation then call the function, to perform Search operations.

Step-11.1:- Declare the necessary 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 and increment count by 1

Step-11.7:- print element not found in the list.

Step-12:- End.

Scanned with Fast Scan

output:-

1. Insert at begining

2. Insert at end

3. Insert at position

4. Delete at i

5. Display from begining

6. search for element

7. exit

enter choice:1

enter value to node:- 3

enter choice:-1

enter value to node:-4

enter choice :-5

linked list element from begining: 4 3

enter choice:-3

enter the position to be inserted: 2

enter value to node:-1

enter choice :- 2

enter value to node : 6

enter choice :- 5

linked list elements from begining : 4 1 3 6

enter choice:- 4

enter position to be deleted : 1

Node deleted.

enter choice : 5

linked list elements from begining : 1 3 6

enter choice : 4

enter position to be deleted : 3

node deleted from the list

enter Choice :- 5

linked list element from begining :- 1 3

enter choice : 6

enter value to be search : 3

Data found in 2 position

enter Choice: 6

enter value to be search:-7

error 7 not found in list

enter choice :-7

program no:- 5

Aim:- Set operations

Algorithm

Step-1:- start

Step-2:- Declare the necessary variable.

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.

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 Cannot perform intersection

Step-5.3:- else read the elements is both the sets.

Step-5.4:- Repeat the step 5.5 to 5.1 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 element in both sets.

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

Scanned with Fast Scan

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 until i<m

Step-7.1:- print c[i]

Step-7.2:- increment i by 1.

Scanned with Fast Scan

output:-

choice to perform

1. union

2. Inter section

3. Difference

4. exit

choice:-1

enter first set: 3

enter second set: 3

enter first set (0|1):1

i

enter set :(0|1) 0 0 1

elements of set 1 union set 2 : 1, 0, 1

Choice to perform:

1. union 2. Inter section 3. Difference 4. exit

Choice: 3

enter first set: 4

enter second set: 4

enter first set :(0|1) 1

0

0

1

enter second set:(0|1) 1

0

1

0

elements of set 1 :- set 2 : 0 0 0 1

choice to perform

1. union 2. intersection 3. Difference 4.exi

choice:- 2

enter first set : 3

enter second set: 3

enter first set :(0|1) 1

0

0

enter second set:(0|1)

1

0

1

elements of set :1

Intersection set 2 1 0 0

Choice to perform

1. union 2. Intersection 3. Difference 4.e

choice : 4

program exit successfully.

program no:- 6

Aim:- Binary search tree

Algorithm

Step-1:- Start

Step-2:- Declare a structure and structure pointers for insertion, deletion and search operations and also declare a function for inorder traversal.

Step-3:- Declare a pointer as root and also the required variables.

Step-4:- Read the choice from the user to perform insertion, deletion and searching and inorder traversal.

Step-5:- if the user choose 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 not root then allocate memory for the root.

Step-5.3:- Set 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 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 not 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 → left 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.

Scanned with Fast Scan

Step-7.1:- Declare the necessary 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 pointers.

Step-8.1:- if root not equals to null recursively call the functions by passing root → left.

Step-8.2:- point root → info

Step-8.3:- call the traversal function recursively by passing root → right.

Scanned with Fast Scan

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:- 10

root is 10.

Inorder traversal of binary tree is: 10

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: 12

root is 10

Inorder traversal of binary tree: 10 12

1. Insert in binary tree

Scanned with Fast Scan

2. Delete from binary tree

3. Inorder traversal of binary tree

4. search

5. excit

enter choice :-1

enter new element : 16

root is 10

Inorder traversal of binary tree: 10 12 16

1. Insert in binary tree

2. Delete from binary tree

3. Inorder traversal of binary tree.

4. search

5. excit

enter choice :1

enter new element : 14

Inorder traversal binary tree : 10 12 14 16

1. Insert in binary tree

2. Delete from binary tree

3. Inorder traversal of binary tree

4. search

5. excit

enter choice:-2

enter the element to be deleted: 15 10 12 14 15

1. Insert in binary tree

2. Delete binary tree

3. Inorder traversal of binary tree

4. search

5. exit

enter choice:-3

Inorder traversal of binary tree is :- 10 12 14

1. Insert in binary tree

2. Delete from binary tree

3. Inorder traversal of binary tree

4. search

5. exit

enter choice:-4

Search operation in binary tree

enter the element to be searched: 16

element 16 which was searched is found and 15-16

1. Insert in binary tree

2. Delete from binary tree

3. Inorder traversal of binary tree

4. search

5. exit

enter choice:-5.