STACK:-

A stack is a collection of elements in which all insertions and deletions are made at one end, called the top of stack.

Algorithm for Push:-

Let 'n' is the data item to be pushed.

1) If TOP=SIZE-1, then
    a. Display "Stack Overflow"
    b. Exit
2) Else
    a. Top=TOP+1
3) A(TOP)=n
4) Exit

Algorithm for Pop:-

1) If TOP<0, then
    a. Display "Stack Underflow"
    b. Exit
2) Else
    a. Remove the top most element.
3) N= A(TOP), delete n
4) TOP=TOP-1
5) EXIT

Queue:-

A queue is collection of elements in which all insertions are made at one end, called rear of the queue and all deletions are made at the other end, called front of the queue.

Algorithm for Enqueue:-

1) Initialise FRONT=0, REAR=-1
2) Input the value to be inserted and assign to variable 'n'
3) If(REAR>=SIZE)
    a. Display "Queue Overflow"
    b. Exit
4) Else
    a. REAR=REAR+1
5) A(REAR)=n
6) Exit

Algorithm for Dequeue:-

1) If (REAR<FRONT)
    a. FRONT=0, REAR=-1
    b. Display "Queue Underflow"
    c. Exit
2) Else
    a. n=A(FRONT)
    b. Delete n
3) FRONT=FRONT+1
4) Exit

F-next
THE NEXT LEVEL OF EDUCATION

Scanned by CamScanner

Defination

Double Ended Queue:

A double ended queue is an abstract data structure that implements a queue for which elements can only be added to or removed from the front (head/left) or back (tail/right). It is also called a head-tail linked list.

The operations that can be performed on dequeues are

- Insert an item from front end
- Insert an item from rear end
- Delete an item from front end
- Delete an item from rear end
- Display the contents of queue.

Circular Queue:-

When a circular queue is implemented, it is thought of an array as a circle rather than a straight line. It means that after reaching at last position of the array, the rear moves towards the first position of the array, only when there is a vacant place at the beginning of the array.

Algorithm for insert element:-

1) If(FRONT=-1)
    a. FRONT=0
    b. REAR=0
2) Else
    a. If (REAR=Size-1)
        i. REAR=0
    b. Else
        i. REAR=REAR+1;
3) A[REAR]=n
4) Exit

Priority Queue:-

A priority queue is a data structure in which elements are inserted arbitrarily but deleted according to their priority. If elements have equal priorities then the usual rule applies, that is first element inserted should be removed first.

Linked List:-

A linked list is an ordered collection of data in which each element contains the location of the next element; that is each element contains two parts: data and link.

a) The data part holds the useful information, the data to be processed.
b) The link is used to chain the data together. It contains a pointer that identifies the next element in the list.

E-next
THE NEXT LEVEL OF EDUCATION

Scanned by CamScanner

Singly Linked List:-

Algorithms:-

1) **Insert a node**

A. Insert a Node at the beginning

i. Input DATA to be inserted
ii. Create a NewNode Temp
iii. Temp -> DATA =n
iv. If (Start = NULL)
    a. Temp -> next = NULL
v. Else
    a. Temp ->Next = Start
vi. Start = Temp
vii. Exit

E-next
THE NEXT LEVEL OF EDUCATION

B. Insert a node at the end

i. Input DATA to be inserted
ii. Create a NewNode Temp
iii. Temp -> DATA =n
iv. Temp -> Next = NULL
v. If (Start = NULL)
    a. Start = Temp
vi. Else
    a. q = Start
    b. While (q -> Next not Equal to NULL)
        q = q -> next
vii. q -> next = Temp
viii. Exit

Scanned by CamScanner

C. **Insert after a particular node**

i. Input the node t after which you want to insert
ii. PTR = Start
iii. While (t not equal to PTR ->DATA AND PTR ->Next!= NULL)
    a. PTR = PTR ->Next
iv. If (t equal to PTR -> DATA)
    a. Create a NewNode temp
    b. temp -> DATA = n
    c. temp -> Next = PTR -> Next
    d. PTR -> Next = temp
v. Else
    a. Display " ELEMENT NOT FOUND"
vi. Exit

E-next
THE NEXT LEVEL OF EDUCATION

D. **Insert before a particular node**

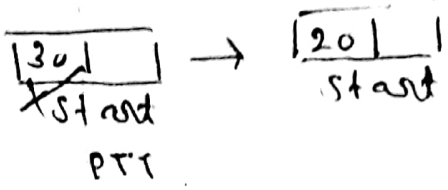
i. Input the node t after which you want to insert
ii. PTR = Start
iii. While (t not equal to PTR ->DATA AND PTR ->Next!= NULL)
    a. PTT = PTR
    b. PTR = PTR ->Next
iv. If (t equal to PTR -> DATA)
    a. Create a NewNode temp
    b. temp -> DATA = n
    c. temp -> Next = PTR -> Next
    d. PTT -> Next = temp
v. Else
    a. Display " ELEMENT NOT FOUND"
vi. Exit

Scanned by CamScanner

2) **Delete a node**

A. **Delete a node at the beginning**

i. PTT = Start
ii. Start = Start -> Next
iii. Delete PTT
iv. Exit



B. **Delete a particular node**

i. Input the node t after which you want to insert
ii. PTR = Start
iii. While (t not equal to PTR ->DATA AND PTR ->Next!= NULL)
    a. PTT = PTR
    b. PTR = PTR ->Next
iv. If (t equal to PTR -> DATA)
    a. PTT -> Next = PTR -> Next
    b. Delete PTR
v. Else
    a. Display " ELEMENT NOT FOUND"
vi. Exit

E-next
THE NEXT LEVEL OF EDUCATION

3) **Displaying all nodes**

A. Temp = Start
B. While (Temp not equal to NULL)
    i. Display Temp -> DATA
    ii. Temp = Temp -> Next
C. Exit

Scanned by CamScanner

Algorithm to concatenate two linked lists:

Concatenate (s1, s2, &s3)

a) Initialise Temp = NULL, PTR = NULL , S3 = NULL
b) While(s1 is not equal to NULL)
    i. Create a NewNode Temp
    ii. Temp -> Data = s1 -> Data
    iii. Temp -> Next = NULL
    iv. s1 = s1 -> Next
    v. If (*s3 equal to Null)
        I. *s3 = NULL
    vi. Else
        I. PTR -> Next = Temp
    vii. PTR = Temp
c) While(s2 is not equal to NULL)
    i. Create a NewNode Temp
    ii. Temp -> Data = s2 -> Data
    iii. Temp -> Next = NULL
    iv. s2 = s2 -> Next
    v. If (*s3 equal to Null)
        I. *s3 = temp
    vi. Else
        I. PTR -> Next = Temp
    vii. PTR = Temp
d) PTR -> Next = NULL
e) Exit

E-next
THE NEXT LEVEL OF EDUCATION

Scanned by CamScanner

4) **Searching a node**

A. Input Search value n
B. PTR = Start
C. While (PTR not equal to NULL)
    I. If (PTR -> DATA equal to n)
        a. Display "Search value found"
        b. Exit
    II. Else
        a. PTR = PTR -> Next
D. If (PTR equal to NULL)
    I. Display " Search value not found"
E. Exit

E-next
THE NEXT LEVEL OF EDUCATION

5) **Count the total number of nodes**

A. Initialise N=0
B. PTR = Start
C. While (PTR not equal to NULL)
    i. N=N+1
    ii. PTR = PTR ->Next
D. Exit

Scanned by CamScanner

6) **Reverse the list**

A. PTR = Start
B. Temp = NULL
C. PTT = PTR -> Next
D. While (PTT not equal to NULL)
    i. Temp = PTR
    ii. PTR = PTT
    iii. PTT = PTR -> Next
    iv. PTR -> Next = Temp
E. Start = Temp
F. Exit

E-next
THE NEXT LEVEL OF EDUCATION

Scanned by CamScanner

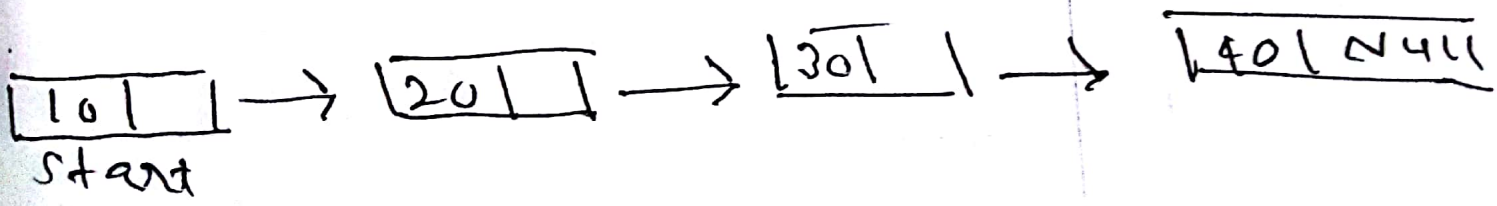
7) **Sort number of elements.**

A. If (Start = NULL)
   i. Display " There are no elements in the list"

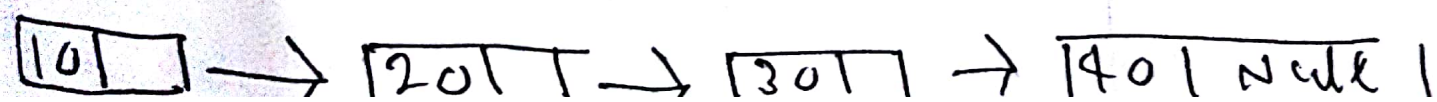
B. Else
   i. PTR = Start
   ii. While (PTR Not equal to NULL)
      a. Temp = PTR -> Next
      b. While (PTR Not Equals to NULL)
         i. If(PTR ->DATA > Temp -> DATA)
            Swap PTR -> DATA with Temp -> DATA
      c. PTR = PTR -> Next

C. Exit

Ernext
THE NEXT LEVEL OF EDUCATION



initialize = Null



Scanned by CamScanner