Pg 1, Farrellin Abdul

Date:

Q.1. Write the definition of the following functions for stack, while re-utilizing the implemented functions of Linked list.

⇒ using functions of linked list:

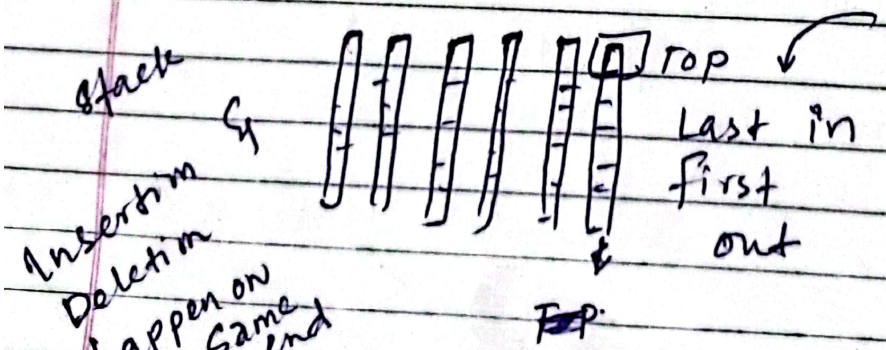
- DECLARE(S)
- Empty(S)
- Top(S)
  ⇒ Retrieve(First(S), S)
- Push(e, S)
  ⇒ Insert(e, First(S), S)
- Pop(S).

Retrieve (First(S), S), Delete (First(S), S).

STACK:

(Special type of list)

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last come first out) OR FIFO (First come last out).



Pg 2 Formula Model

Date:

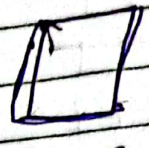
Declare (S):

Declare: → S:

The function value of Declare(S) is an empty stack.

Empty: → S:

The function Empty(S) causes the stack to be emptied and returns the position which is End(S)



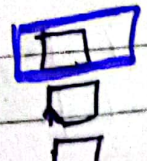
Along with Empty there is another function IsEmpty(S) which is a boolean function which returns true is the stack is empty, else return false.

Top(S):

Top: S → E:

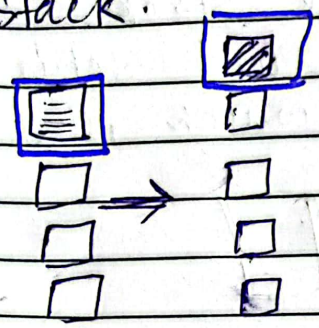
This function is defined to return the first element in the list but if the list is empty, then its value is undefined.

Top(S):



Push(e, s):

Push: E X S → L:
Push(e, s)
This function takes two arguments e, and S, element and stack which insert element (e) at the top of the stack.

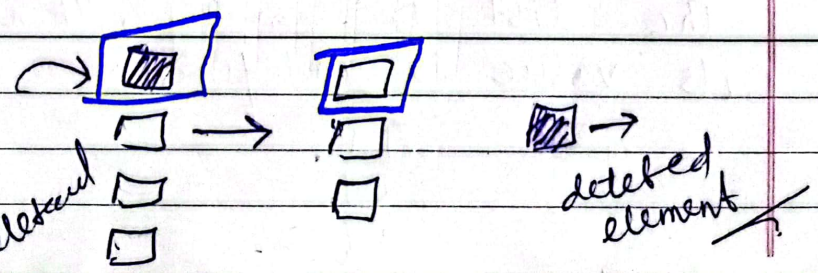


~~Push(e, s)~~

Pop(s):

Pop: S → E:
Pop(s):

This function removes the top element from the stack. i.e, return the top element and delete it from the stack:



Pg: 04 Fortune Edu

Date

Implementation of stack in cpp using Linked Lists:

```
#include <iostream>
using namespace std;
typedef struct Node
{ int data;
    Node *next;
} Node;
Class Stack
{
Private:
  Node *top;
  int Max-Size;
  int size=0;
Public:
  void Declare (int size)
{     Max-size= size;
      top =NULL;
      Cout<< "Stack Created. \n";
}
void Push (int data)
{
  if (size>= Max-size)
    { cout<< "Stack is full" <<endl; }
  else
  { Node *temp;
      temp= new Node();
      temp-> data= data;
      temp -> next=top;
      top= temp;
      size ++;
    Cout<< "Element pushed into The Stack\n";
  }  }
```

Pg 5 Farmulla Adeel

Date:

```
void pop()
{
  if(size==0)
    { cout<<"Stack is empty"<<endl;}
  else
  { Node *temp;
    temp = top;
    top = temp->next;
    temp->next = NULL;
    delete temp;
    size--;
    cout<<"Element removed from Stack\n";
  }
}
void Top()
{
  if(size==0)
  {cout<<"Stack is empty\n";}
  else
  {
    cout<<"Top element is= "<<top->data<<endl;
  }
void size()
  {
  cout<<"The size of the stack is="<<size<<endl;
  }
void Empty()
{ Node *temp;
  temp = top;
    delete temp;
    top= NULL;
    size =0
cout<<"Stack emptied\n"; }
```

Pg 6. Formula Adul.

Date:

```
Void display ()
{
 if(size==0)
{
cout<< " stack is empty \n";
}
else
{
 Node * temp;
  temp=top;
 while(temp!=NULL)
  {
   cout<< temp->data<<" ";
   temp= temp->next;
  }
 }
}
};
 int main()
 {
 int choice, item, s;
  Stack S;
 while(1)
 {
  cout<< " Stack operations "<<endl;
  cout<<"1. Declare Stack "<<endl;
  cout<<"2. Push Element "<<endl;
  cout<<"3. Pop Element "<<endl;
  cout<<" 4. Display Stack "<<endl;
  cout<<"5. Empty the Stack "<<endl;
  cout<<"6. Size of Stack "<<endl;
  cout<<"7. Top item "<<endl;
  cout<<"8. Quit \n 9. Enter your choice \n";
```

Pg7 Fanth Nawaz Date:

```
cin>> choice;
switch(choice)
{ case 1:
    cout<< "Enter size of stack ";
    cin>>s;
    s.Declare(s);
    break;
case2:
cout<< "Enter value to be pushed ";
  cin>>item;
  s.Push(item);
  break;
Case 3:
    s.pop();
    break;
case 4:
    s.Display();
    break;
Case 5;
    s.Empty();
    break;
case 6;
  s.Size();
  break;
Case7: s.top(); break;
case8: return 0;
    break;
default:
    cout<< "Wrong choice"<<endl;
}
}
return 0;
}
```