WRITE AN ALGORITHM AND PROGRAM FOR THE FOLLOWING:

1. IMPLEMENT STACK AS LINKED LIST

ALGORITHM:

1. Create a linked list with first node as null pointer.
2. When an item is pushed in stack add a new node in the beginning of the linked list
3. If an item is to be popped from the stack, check if the list is empty, if so, throw UNDERFLOW and terminate process
4. If list is not empty then remove the item at the beginning of the linked list (i.e. the top of the stack).
5. Display the item stored in the first node of the list when peek ( ) is called.

Program Code

#include <iostream>

#include "List.h"

using namespace std;

template <class T>

class Stack

{

const int maxM;

List<T> l;

public:

Stack(int s):maxM (s)

{

l.setMax(maxM);

}

void push(T data)

{

l.insertAtFront(data);

}

T pop()

{

T temp=l.removeFromFront();

return temp;

}

void peek()

{

cout<<l.peekFront();

}

/\* void display()

{

l.display();

}\*/

int stackSize()

{

return l.sizeG();

}

};

int main()

{

int op,sizeSt,num;

cout<<"Size?:";

cin>>num;

sizeSt=num;

Stack<int> s(num);

while(1)

{

try

{

cout<<"\nEnter Operation:\t1)Push\t2)Pop\t3)Exit:";

cin>>op;

if(op==1)

{

if(s.stackSize()<sizeSt)

{

cout<<"\nEnter number to be pushed:";

cin>>num;

s.push(num);

//cout<<"\nData is stack:";

//s.display();

cout<<"\nTop of Stack:\t";

s.peek();

}

else

{

throw(List<int>::OVERFLOW());

}

}

else if(op==2)

{

s.pop();

//cout<<"\nData is stack:";

//s.display();

cout<<"\nTop of Stack:\t";

s.peek();

}

else

{

exit(0);

}

}

catch(List<int>::UNDERFLOW)

{

cout<<"\nEmpty Stack!";

}

catch(List<int>::OVERFLOW)

{

cout<<"\nStack FULL!";

}

}

return 0;

}

2. IMPLEMENT QUEUE AS LINKED LIST

ALGORITHM:

1. Create a linked list with first node as null pointer.
2. When an item is enqueued in queue add a new node in the beginning of the linked list and store the item in that node.
3. If an item is to be dequeued from the queue, check if the list is empty, if so, throw UNDERFLOW and terminate process
4. If list is not empty then remove the item at the end of the linked list.

i.e traverse list until presnetNode->next!=null pointer and then set next of node before presentNode to null pointer.