**Godavari College Of Engineering, Jalgaon.**

**Subject Name:** Data Structure **Teacher Name:** Prof.S.S.Shete

**Practical No**. : 6 **Date:**

**Class: S**.E **Roll No:**

**Title:**  Write a program to implement a stack using two queues such that the push operation runs in constant time and the pop operation runs in linear time( O(n)).

**Aim:** To implement a stack using two queues such that the push operation runs in constant time and the pop operation runs in linear time.

**Theory:**

This method makes sure that newly entered element is always at the front of ‘q1’, so that pop operation just dequeues from ‘q1’. ‘q2’ is used to put every new element at front of ‘q1’.

1. push(s, x) operation’s step are described below:

* + - Push x to q2
    - One by one pop everything from q1 and enqueue to q2.
    - Swap the names of q1 and q2

2. pop(s) operation’s function are described below:

* + - Pop an item from q1 and return it.

**Program:-**

#include<stdio.h>

#include<bits/stdc++.h>

using namespace std;

struct Stack

{

queue<int> q1,q2;

void push(int x)

{

if(q1.empty())

{

q2.push(x); //EnQueue operation using STL

}

else

{

q1.push(x); //EnQueue operation using STL

}

}

int pop()

{

int count,size,item;

if(q2.empty())

{

size=q1.size(); //size=no of elements;

count=0;

while(count<size-1)

{ //transfering n-1 elements

q2.push(q1.front()); // DeQueue operation using STL

q1.pop();

count++;

}

item=q1.front();

q1.pop();

return item; //popping out the element

}

else

{

size=q2.size();

count=0;

while(count<size-1){

q1.push(q2.front());

q2.pop();

count++;

}

item=q2.front();

q2.pop();

return item;

}

}

};

int main()

{

Stack s;

int x,count=0;

printf("implementing stack with two queues");

printf("enter any integer to push and 0 to stop pushing :- ");

scanf("%d",&x);

while(x)

{

s.push(x);

scanf("%d",&x);

count++;

}

printf("\n\n===================================================\n\n");

printf("\n\n\t now popping.......\n\n");

while(count)

{

cout<<s.pop()<<endl;

count--;

}

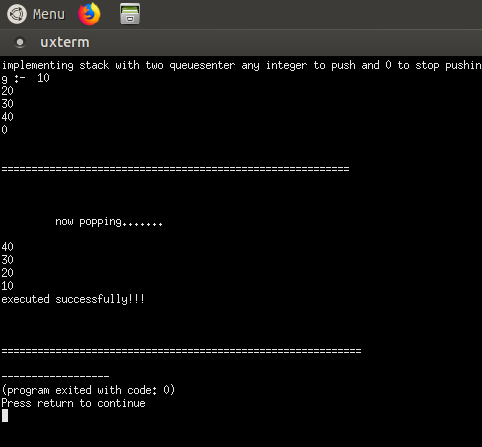
printf("executed successfully!!! \n\n");;

printf("\n\n=====================================================");

return 0;

}

**Output:**

****

**Conclusion:-**