IMPLEMENT STACK USING TWO QUEUES

We have seen the code to implement a Queue using two stacks. This question is opposite of that. Use only Queue data structure to implement a Stack.

The solution to this is similar to the previous post. We will use two Queues, Insertion into Stack (push operation) will insert element in one Queue and Pop operation (removing element from Stack) will remove the data from another Queue. Let's name the Queues as Queue1 and Queue2.

Ideally both push and pop operations should be constant time operations (O(1)). But that is the case when Stack is implemented using either Array or Linked List data structure.

But if we are implementing Stack using Queue data structure, then both the operations cannot be constant time operations. At least one of the operations has to be expensive.

It assumes the Queue to be implemented as shown in this post

The class definition will look something like below:

```
1    class SpecialStack
2    {
3    public:
```

```
4         void push (int data);
5         int pop();
6
7     private:
8         Queue queue1;
9         Queue queue1;
10    };
```

Ideally such classes should be templates because we don't want our Stack or Queue to be restricted to store int data types only. So the production code generally is like:

```
1     template  class Stack
2     {
3     public:
4         void push (T data);
5         T pop();
6
7     private:
8         Queue queue1;
9         Queue queue1;
10    };
```

But we will stick to the previous definition only for sake of simplicity.

Method-1: Push is constant time but Pop operation is O(n) time

```
1    void SpecialStack::push(int data){
2        queue1.enqueue(data)
3    }
4
5    int SpecialStack::pop(){
6        int returnValue =-1; // indicate Stack Empty.
7
8        while(!queue1.isEmpty())
9        {
10           returnValue = queue1.dequeue();
11
12           // If it was last element of queue1. return it.
13           if(queue1.isEmpty())
14               break;
15           else
16               queue2.enqueue(returnValue);
17       }
18
19       // swap the names of queue1 and queue2.
20       // If swapping is not possible then we will have to move all the elements from queue2 to
21   queue1
```

```
22        // or have another flag to indicate the active queue.
23        Node * temp = queue1;
24        queue1 = queue2;
25        queue2 = temp;
26
27        return returnValue;
      }
```

Method-2: Pop is constant time but Push operation is O(n) time

```
1     void SpecialStack::push(int data){
2        queue2.enqueue(data);
3
4        while(!queue1.isEmpty()){
5           queue2.enqueue(queue1.dequeue());
6        }
7
8        // swap the names of queue1 and queue2.
9        // If swapping is not possible then we will have to move all the elements from queue2 to
10    queue1
11       // or have another flag to indicate the active queue.
12       Node * temp = queue1;
13       queue1 = queue2;
14       queue2 = temp;
```

```
15      }
16
17      // Put proper check to see if no element in Queues
18      int SpecialStack::pop(){
19        return queue.dequeue();
        }
```

Source: http://www.ritambhara.in/implement-stack-using-two-queues/