Implementation of queue using stack

2 stacks are required.

Aprroach 1 :

- push all the elements from in orignal stack to temp stack
- push the current elem to orig stack
- copy all the elem from temp stack to orig stack.

Main motive is to get the top element which is done by pushing element to original stack.

```
class MyQueue {
public:
    stack<int> s1;
    stack<int> s2;
    void push(int x) {
        while(!s1.empty())
        {
            s2.push(s1.top());
            s1.pop();

        }
        s1.push(x);
        while(!s2.empty())
        {
            s1.push(s2.top());
            s2.pop();

        }
    }

    int pop() {
        if(s1.size()==0)
            return -1;
       int ans=s1.top();
       s1.pop();
       return ans;
    }

    int peek() {
        if(s1.size()==0)
            return -1;
       int ans=s1.top();
       return ans;
    }

    bool empty() {
        if(s1.size()==0)
            return 1;
        return 0;
    }
};

/**
 * Your MyQueue object will be instantiated and called as such:
 * MyQueue* obj = new MyQueue();
 * obj->push(x);
 * int param_2 = obj->pop();
 * int param_3 = obj->peek();
 * bool param_4 = obj->empty();
 */
```

Using 2 stack but reducing time complexity from O(n) to amortized complexity O(1).

Amortized means in most of the cases the complexity will be O(1) but in some cases it can be O(n) i.e. when a pop element is called

```
class MyQueue {
public:
```

```
    stack<int> inp;
    stack<int> out;
    void push(int x)
    {
       inp.push(x);
    }

    int pop() {
        if(out.empty())
        {
            while(!inp.empty())
            {
                out.push(inp.top());
                inp.pop();
            }
        }
        int ans=out.top();
        out.pop();
        return ans;
    }

    int peek() {
       if(out.empty())
        {
            while(!inp.empty())
            {
                out.push(inp.top());
                inp.pop();
            }
        }
        return out.top();
    }

    bool empty() {
        if(inp.size()==0 && out.size()==0)
            return 1;
        return 0;
    }
};

/**
 * Your MyQueue object will be instantiated and called as such:
 * MyQueue* obj = new MyQueue();
 * obj->push(x);
 * int param_2 = obj->pop();
 * int param_3 = obj->peek();
 * bool param_4 = obj->empty();
 */
```