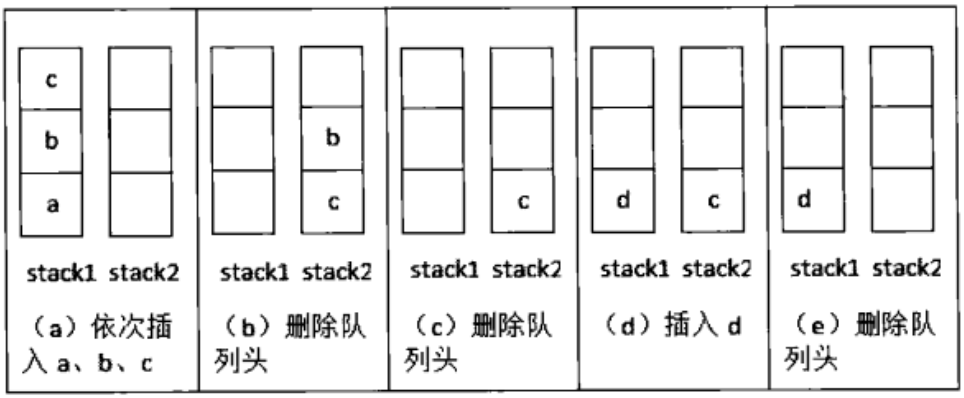


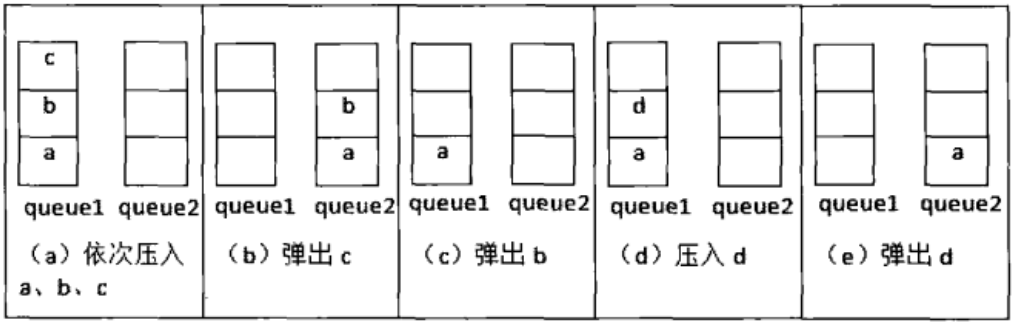
用两个栈模拟一个队列的操作

import java.util.Stack;

public class Demo07 {

```
Stack<Integer> stack1 = new Stack<Integer>();
Stack<Integer> stack2 = new Stack<Integer>();

public void push(int node) {
    stack1.push(node);
}

public int pop() {
    if(stack2.size()<=0){
        while(stack1.size()>0){
            //弹出stack1的栈顶元素压入stack2
            stack2.push(stack1.pop());
        }
    }

    if(stack2.isEmpty()){
        try {
            throw new Exception("queue is empty.");
        } catch (Exception e) {
        }
    }

    int head = stack2.pop();
    return head;

}
```

}



用两个队列模拟一个栈的操作

import java.util.ArrayDeque;

import java.util.Queue;

public class Demo08 {

```
Queue<Integer> queue1 = new ArrayDeque<>();
Queue<Integer> queue2 = new ArrayDeque<>();

public void push(int node) {
    //两个栈都为空时，优先考虑queue1
    if (queue1.isEmpty()&&queue2.isEmpty()) {
        queue1.add(node);
        return;
    }

    //如果queue1为空，queue2有元素，直接放入queue2
    if (queue1.isEmpty()) {
        queue2.add(node);
        return;
    }

    if (queue2.isEmpty()) {
        queue1.add(node);
        return;
    }

}

public int pop() {
    //两个栈都为空时，没有元素可以弹出
```

```
        if (queue1.isEmpty()&&queue2.isEmpty()) {
            try {
                throw new Exception("stack is empty");
            } catch (Exception e) {
            }
        }
        //如果queue1为空，queue2有元素，
        //将queue2的元素依次放入queue1中，直到最后一个元素，我们弹出。
        if (queue1.isEmpty()) {
            while (queue2.size()>1) {
                queue1.add(queue2.poll());
            }
            return queue2.poll();
        }

        if (queue2.isEmpty()) {
            while (queue1.size()>1) {
                queue2.add(queue1.poll());
            }
            return queue1.poll();
        }

        return (Integer) null;
    }

    public static void main(String[] args) {
        Demo08 demo08 = new Demo08();
        demo08.push(1);
        demo08.push(2);
        demo08.push(3);
        demo08.push(4);
        System.out.println(demo08.pop());
        System.out.println(demo08.pop());
        demo08.push(5);
        System.out.println(demo08.pop());
        System.out.println(demo08.pop());
        System.out.println(demo08.pop());
    }
```

}