1

题目：
给定一个栈S，保存的数据类型均为int，其仅具有如下三个运算：

- S.push(x)：将x压入栈中
- S.pop()：将栈顶元素出栈，并返回该元素
- S.top()：返回栈顶元素

请你在该栈的基础上，不添加新的数据结构，使得S可以仍然支持上述三种运算（可以重写），并实现一种新的运算：

- S.getMin()：返回栈中最小元素

要求：S的所有操作时间复杂度为O(1)。无需考虑特殊情况如栈空或者栈满等。请给出所有新运算的伪代码。

```
class S {
    int min_element; //最小元素

    S() : min_element(0x7fffffff) { //初始化为max_int
        ...
    } //构造函数内容不变

    void push(int x) {
        min_element = min(min_element, x);
        ... //后面不变
    }

    int getMin() {
        return min_element;
    }
}
```

2

题目：编号为 1, 2, ..., n的 n 辆火车顺序开进栈式结构的站台。请问开出车站的顺序有多少种可能？请写出你的推导过程。

答:

设 $S(n)$ 表示 n 辆火车的出站顺序数量。考虑第一辆进站的火车（编号为 1）在出站序列中的位置。假设火车 1 在第 k 个位置出站（其中 $1 \leq k \leq n$)。则：

- 在火车 1 出站之前，有 $k-1$ 辆火车出站。可能性种数为 $S(k-1)$。
- 在火车 1 出站之后，剩余的 $n-k$ 辆火车（编号为 $k+1$ 到 n）进站并出站，其出站顺序数量为 $S(n-k)$。

因此，对于固定的 k，出站顺序数量为 $S(k-1) \times S(n-k)$。对 k 从 1 到 n 求和，得到递推关系：

$$S(n) = \sum_{k=1}^{n} S(k-1)S(n-k)$$

其中，定义 $S(0) = 1$（表示没有火车时，出站序列为空序列，有一种可能)。

该递归关系对应于卡特兰数的递推定义。卡特兰数的通项公式为：

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

因此，n 辆火车的出站顺序数量为：

$$S(n) = C_n = \frac{1}{n+1}\binom{2n}{n}$$

3

题目:
在文本编辑器中，有一个“撤销（Undo)”和“恢复（Redo)”的功能。假设编辑器使用两个栈来实现：

- 栈 S1 保存已经执行的操作（每次新操作都会压入 S1)；
- 栈 S2 保存被撤销的操作（每次执行一次撤销，就把 S1 的栈顶弹出并压入 S2；每次执行一次恢复，就把 S2 的栈顶弹出并压入 S1)。

现在给定一系列操作指令（操作包括：

- do x ： 执行一个新操作 x；
- undo ： 撤销一步操作；

- redo： 恢复一步操作)，

请回答：

1. 用栈的基本操作（push、pop、empty）描述**撤销**和**恢复**的实现过程。
2. 如果一开始编辑器为空，依次执行以下操作序列：

> do A, do B, do C, undo, do D, undo

请问最后 S1 和 S2 中分别保存哪些操作?

答:

1. 第一问

undo :

```
op = S1.pop();
S2.push(op);
```

redo :

```
op = S2.pop();
S1.push(op);
```

1. 第二问

> S1 : A,B (B为栈顶元素)
> S2 : C,D (D为栈顶元素)

4

题目:
已知队列的三个基本操作定义如下：

- enQueue(Q, x)： 将元素 x 入队列 Q（在队尾插入)。
- deQueue(Q)： 队列 Q 的队首元素出队，并返回该元素。
- isEmpty(Q)： 判断队列 Q 是否为空。

请你设计一种方法，用**两个普通队列**（Q1, Q2）来实现栈（Stack）的三个基本操作：

- push(x)： 将元素压入栈顶。
- pop()： 弹出并返回栈顶元素。
- isEmpty()： 判断栈是否为空。

要求：

1. 给出算法思路，并写出 push 和 pop 的伪代码。
2. 分析 push 和 pop 操作的时间复杂度。
3. 若依次执行以下操作序列：

> push(1), push(2), push(3), pop(), push(4), pop(), pop()

请写出每次 pop() 的返回结果。

答 :

1. ***算法思路和代码***

- 用 *Q1* 保存栈中的所有元素, *Q1* 的队首元素是栈顶元素
- 执行 *push(x)* 操作时:
  i. 将 *x* 入队到 *Q2*
  ii. 将 *Q1* 中所有元素依次出队并入队到 *Q2*
  iii. 交换 *Q1* 和 *Q2*
- 执行 *pop()* 操作时: 直接从 *Q1* 出队并返回该元素
- 执行 *isEmpty()* 操作: 判断 *Q1* 是否为空

```
void push(item x) {
    enQueue(Q2, x);
    while (!isEmpty(Q1)) {
        enQueue(Q2, deQueue(Q1));
    }
    swap(Q1, Q2);
}

item pop() {
    return deQueue(Q1);
}
```

2. ***时间复杂度***

push : *O(n)*
pop : *O(1)*

3. ***返回结果***

> 3, 4, 2