1

直接给出 C++ 实现代码：

```
template <typename T>
class StackQueue{
private:
    std::stack<T> stk, temp;
public:
    void enqueue(const T& val){
        stk.push(val);
    }
    void dequeue(){
        while (!stk.empty()){
            temp.push(stk.top());
            stk.pop();
        }
        temp.pop();
        while (!temp.empty()){
            stk.push(temp.top());
            temp.pop();
        }
    }
    bool queue_empty(){
        return stk.empty();
    }
};
```

我使用栈 `stk` 存储队列中元素，入队就是入栈，队空就是栈空. 唯一稍麻烦的是出队，需要将 `stk` 中元素依次出栈并压入另一个临时栈 `temp`，此时 `temp` 中元素是 `stk` 中元素的倒序，删除 `temp` 的栈顶元素（即 `stk` 的栈底元素，即队首元素），再将 `temp` 中元素依次出栈并压入 `stk`，就完成了出队操作.

这种出队实现是 $O(n)$ 的，效率很低，无法真正投入使用. 我并不知道如何用栈 $O(1)$ 地实现这个操作.

一个典型的队列还应当能查看队首元素、队尾元素和元素数目，既然题目没有要求，我也就没有实现. 如果要实现的话，队尾元素就是栈顶元素，队列大小就是栈大小，但查看队首仍要用到 `temp`，是 $O(n)$ 的，一种可能的优化是在出队的同时记录新队首元素.

由于使用了 `std::stack`，我没有使用题目提供的函数原型. 当然，这并没有什么事实上的区别.

我认为我的代码已经足够清晰了，因此没有写注释.

2

我们来证明答案是 $C_{2n}^n/(n+1)$.

把入栈看作 1，出栈看作 0，原问题显然等价于：将 n 个 1 和 n 个 0 排成一个 2n 长的二进制序列，使其任意前缀（即一个数位及其前面的所有数位）中 0 的个数不超过 1 的个数，求这样的合法序列有多少种.

如果随意排列，可能情况数是 C_{2n}^n，因此只需证明不合法情况数为 $\frac{n}{n+1}C_{2n}^n$，即 C_{2n}^{n-1} 种.

注意到，由 n-1 个 1 和 n+1 个 0 组成的二进制数恰好有 C_{2n}^{n-1} 种，这启发我们，如果能建立这样的二进制数与不合法序列之间的一一映射，就能完美优雅地解决问题.

我们这样构造这个映射：

- 对于一个不合法序列，找到第一个不合法数位，即前缀中 0 的个数比 1 的个数多 1 的数位.
- 设这一数位的前缀中有 m 个 1，m+1 个 0，即这一数位为第 2m+1 位.
- 那么，在之后的 2n-2m-1 位中，一定存在 n-m 个 1，n-m-1 个 0.
- 翻转这后 2n-2m-1 位中的 0 和 1，翻转后的数中就有 n-1 个 1 和 n+1 个 0.

先证明这是一个单射，即不同不合法序列的像不同：

- 两个不同不合法序列 x_1, x_2 若第一个不合法数位不同，则其像必定不同. 这是因为，设这两个序列的第一个不合法数位分别为 a, b，不妨设 $a < b$，那么 x_1 的 a 数位前缀是不合法的，x_2 的 a 数位前缀是合法的，而映射不改变这前 a 位，因此其像不同.
- 两个不同不合法序列若第一个合法数位相同，则其像也必定不同. 这是因为翻转是双射.

再证明这是一个满射，即任意一个由 n-1 个 1 和 n+1 个 0 组成的二进制数存在原像：

- 这个数一定存在某一奇数位 2p+1，其前缀中有 p 个 1 和 p+1 个 0.
- 翻转后 2n-2p-1 位中的 0 和 1，翻转后的数有 n 个1和 n 个0，因为前 2p+1 位没有改变，因此所得到序列是不合法的.

至此，我们建立了一一映射，也就证明了答案是 $C_{2n}^n/(n+1)$.

3

题目的描述对我来说很别扭，我习惯于这样的叙述：将 n 个元素按照输入次序编号为 $1, 2, \cdots, n$，考虑映射 P，P 将元素 i 映射到 i 出栈的位次 $P(i)$. 求证输出序列合法的充分必要条件是：不存在元素 $i < j < k$，使 $P(k) < P(i) < P(j)$.

显然这与原问题等价，下面证明这个命题.

必要性：

$\forall i < j < k$，若 $P(k) < P(i) < P(j)$，说明 k 出栈时 i, j 都没有出栈，由于它们是按顺序依次入栈的，出栈时必然有 $P(j) < P(i)$，矛盾.

充分性：

下面对序列元素数 m 归纳.

$m = 3$ 时，直接枚举所有排列即可.

假设 $\forall m \leq n$ 时都成立，$m = n + 1$ 时，设 $P(t) = n + 1$.

$\forall i < t < j$，由条件有 $P(i) < P(j)$，即所有编号小于 t 的元素在所有编号大于 t 的元素之前出栈. 因此，这个过程必然是：所有编号小于 t 的元素入栈出栈，t 入栈，所有编号大于 t 的元素入栈出栈，t 出栈.

由归纳假设，这两个子过程序列都是合法的，因此整个序列是合法的.

特别地，若 $t = 1$ 或 $t = n + 1$，也可以作类似讨论，其成立性容易证明.

由数学归纳法，证明完毕.