考试 2（栈/队列）

学号末两位数+3 按 6 取模，值为 1 的同学做第一和第二部分的第 1 题（共 2 题），以此类推，值为 0 的同学做第一和第二部分的第 6 题（共 2 题）。

过程提交每小时一次，**每次持续5分钟**，提交文件夹为“考试过程提交”文件夹中的相应班级，过程源代码压缩包命名：学号_姓名_时间，如 2017000123_张三_14点.rar

最终代码提交截止时间为 17:35，不延时，请提前做好准备，提交文件夹为“考试提交”文件夹中的相应班级，源代码压缩包命名：学号_姓名_考试 2

解题请使用栈/队列操作(必须自己声明实现)，未使用一律 0 分。

（**注：**文件未打压缩包、命名格式不规范、提交到错误文件夹，一律不批改。）

第一部分

1）You are given two arrays (without duplicates) nums1 and nums2 where nums1's elements are subset of nums2. Find all the next greater numbers for nums1's elements in the corresponding places of nums2. The Next Greater Number of a number x in nums1 is the first greater number to its right in nums2. If it does not exist, output -1 for this number.

Example 1:

```
Input: nums1 = [4,1,2], nums2 = [1,3,4,2].
Output: [-1,3,-1]
Explanation:
    For number 4 in the first array, you cannot find the next greater
number for it in the second array, so output -1.
    For number 1 in the first array, the next greater number for it in the
second array is 3.
    For number 2 in the first array, there is no next greater number for it
in the second array, so output -1.
```

Example 2:

```
Input: nums1 = [2,4], nums2 = [1,2,3,4].
Output: [3,-1]
Explanation:
    For number 2 in the first array, the next greater number for it in the
second array is 3.
    For number 4 in the first array, there is no next greater number for it
in the second array, so output -1.
```

Note:

1. All elements in `nums1` and `nums2` are unique.
2. The length of both `nums1` and `nums2` would not exceed 1000.

2）We are given an array asteroids of integers representing asteroids in a row. For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed. Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are the same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:

```
Input:
asteroids = [5, 10, -5]
Output: [5, 10]
Explanation:
The 10 and -5 collide resulting in 10.  The 5 and 10 never collide.
```

Example 2:

```
Input:
asteroids = [8, -8]
Output: []
Explanation:
The 8 and -8 collide exploding each other.
```

Example 3:

```
Input:
asteroids = [10, 2, -5]
Output: [10]
Explanation:
The 2 and -5 collide resulting in -5.  The 10 and -5 collide resulting in
10.
```

Example 4:

```
Input:
asteroids = [-2, -1, 1, 2]
Output: [-2, -1, 1, 2]
Explanation:
The -2 and -1 are moving left, while the 1 and 2 are moving right.
Asteroids moving the same direction never meet, so no asteroids will meet
each other.
```

Note:

- The length of `asteroids` will be at most `10000`.
- Each asteroid will be a non-zero integer in the range `[-1000, 1000].`.

3）Given a sequence of n integers a_1, a_2, ..., a_n, a 132 pattern is a subsequence a_i, a_j, a_k such that **i** < **j** < **k** and $a_i < a_k < a_j$. Design an algorithm that takes a list of n numbers as input and checks whether there is a 132 pattern in the list.

Note: n will be less than 15,000.

Example 1:

```
Input: [1, 2, 3, 4]
Output: False
Explanation: There is no 132 pattern in the sequence.
```

Example 2:

```
Input: [3, 1, 4, 2]
Output: True
Explanation: There is a 132 pattern in the sequence: [1, 4, 2].
```

Example 3:

```
Input: [-1, 3, 2, 0]
Output: True
Explanation: There are three 132 patterns in the sequence: [-1, 3, 2], [-1,
3, 0] and [-1, 2, 0].
```

4）Implement the following operations of a stack using queues.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- empty() -- Return whether the stack is empty.

Notes:

- You must use *only* standard operations of a queue -- which means only `push to back`, `peek/pop from front`, `size`, and `is empty` operations are valid.
- Depending on your language, queue may not be supported natively. You may simulate a queue by using a list or deque (double-ended queue), as long as you use only standard operations of a queue.
- You may assume that all operations are valid (for example, no pop or top operations will be called on an empty stack).

5）Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Example:

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin();   --> Returns -3.
minStack.pop();
minStack.top();      --> Returns 0.
minStack.getMin();   --> Returns -2.
```

6）Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. The brackets must close in the correct order, "()" and "()[]{}" are all valid but "(]" and "([)]" are not.

第二部分

1）Given a list of daily temperatures, produce a list that, for each day in the input, tells you how many days you would have to wait until a warmer temperature. If there is no future day for which this is possible, put 0 instead. For example, given the list temperatures = [73, 74, 75, 71, 69, 72, 76, 73], your output should be [1, 1, 4, 2, 1, 1, 0, 0].

Note: The length of `temperatures` will be in the range `[1, 30000]`. Each temperature will be an integer in the range `[30, 100]`.

2）Given a non-negative integer *num* represented as a string, remove *k* digits from the number so that the new number is the smallest possible.

Note:

- The length of *num* is less than 10002 and will be ≥ *k*.
- The given *num* does not contain any leading zero.

Example 1:

```
Input: num = "1432219", k = 3
Output: "1219"
Explanation: Remove the three digits 4, 3, and 2 to form the new number
1219 which is the smallest.
```

Example 2:

```
Input: num = "10200", k = 1
Output: "200"
Explanation: Remove the leading 1 and the number is 200. Note that the
output must not contain leading zeroes.
```

Example 3:

```
Input: num = "10", k = 2
Output: "0"
Explanation: Remove all the digits from the number and it is left with
nothing which is 0.
```

3）Evaluate the value of an arithmetic expression in Reverse Polish Notation. Valid operators are +, -, *, /. Each operand may be an integer or another expression.

Some examples:

```
  ["2", "1", "+", "3", "*"] -> ((2 + 1) * 3) -> 9
  ["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6
```

4）Given a string which contains only lowercase letters, remove duplicate letters so that every letter appear once and only once. You must make sure your result is the smallest in lexicographical order among all possible results.

Example:
Given `"bcabc"`
Return `"abc"`
Given `"cbacdcbc"`
Return `"acdb"`

5）Implement a basic calculator to evaluate a simple expression string. The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, **non-negative** integers and empty spaces. You may assume that the given expression is always valid.

Some examples:

```
"1 + 1" = 2
" 2-1 + 2 " = 3
"(1+(4+5+2)-3)+(6+8)" = 23
```

6）Implement the following operations of a queue using stacks.

- push(x) -- Push element x to the back of queue.
- pop() -- Removes the element from in front of queue.
- peek() -- Get the front element.
- empty() -- Return whether the queue is empty.

Notes:

- You must use *only* standard operations of a stack -- which means only `push to top`, `peek/pop from top`, `size`, and `is empty` operations are valid.
- Depending on your language, stack may not be supported natively. You may simulate a stack by using a list or deque (double-ended queue), as long as you use only standard operations of a stack.
- You may assume that all operations are valid (for example, no pop or peek operations will be called on an empty queue).