CSCI 2100A Data Structures

Midterm Examination (Programming Part)
6:30 p.m. - 9:30 p.m., Thursday, April 2, 2015

Room 924, Ho Sin Hang Engineering Building

Instructions

1. The programming midterm is an open-book and open-notes examination. You may bring what you can carry on printed (hard copy) materials. You MUST not take anything that can record program code electronically to the examination venue. You will not need a calculator for any calculation.

2. The operation system will be Ubuntu. The computer configuration will have these basic editors: vi/vim, emacs, gedit, and nano. This system includes GDB for debugging.

3. You can use all the functions provided by standard C library as long as you've included the corresponding header file.

4. The examination will begin when the Chief TA starts the clock and will end when the Chief TA stops the clock, which is usually three hours after the starting time including any missing time due to technical or other difficulties.

5. You are suggested to work on Problem A first and then others afterwards. They are in increasing difficulties from the perspective of algorithm as judged by the instructors.

6. You MUST complete at least one problem in order not to fail the course.

7. Anyone who attempts to spam the server either through excessive submissions, allocating large amount of unnecessary memory, etc. will be penalized severely.

8. Please switch your mobile phones to silent mode and place it under your seat, you are not allowed to use them during the exam.

9. If you want to go to the bathroom, please ask the TAs for permission first.

10. If you leave early from the examination without informing the TAs, you will not be able to come back to the examination.

Problem A - Palindrome String

You are given a non-empty string containing only numbers and English alphabets. Your are asked to decide whether the string is a palindrome or not.

A string is a palindrome if and only if it reads the same backward or forward. We do not distinguish lowercase and uppercase letters in this problem (e.g., B and b are same).

Input There are several test cases in the input file. The fist line is an integer $t \leq 20$, which indicates the number of test cases.

For each test case, there is only one line containing one string with length no greater than 300.

Output For each test case, output one line: "Yes" if the string is a palindrome; "No" otherwise.

Sample Input

```
2
abBa
abcd
```

Sample Output

```
Yes
No
```

Problem B - Majority

Given an array of size n, find the majority element. The majority element is the element that appears more than $\lfloor n/2 \rfloor$ times, where $\lfloor x \rfloor$ returns the maximum integer not greater than x.

You may assume that the array is non-empty and the majority element always exists.

Input There are several test cases in the input file. The first line is an integer $t \leq 20$, which indicates the number of test cases.

For each test cases, there are two lines. The fist line contains an integer $1 \leq n \leq 1000$. The next line contains n positive integers in the range of data type `long`, separated by a space.

Output For each test case, output the majority in one line.

Sample Input

```
2
5
1 2 3 2 2
6
3 4 1 4 4 4
```

Sample Output

```
2
4
```

Source LeetCode Online Judge - Problem 169

Problem C - Max-modify Heap

Design a Max-modify Heap data structure supporting the following operations: `find-max`, `insert`, `delete-max`, and `change-max`, which is to change the value of the maximum element.

Input The first line contains an integer $1 \leq n \leq 1000000$, which indicates the number of operations. The next n lines each contains an operation. There are four different operations in total:

- 1 x: Insert a non-negative integer x into the heap.
- 2: Delete the maximum element.
- 3: Return the maximum element of the heap.
- 4 v: change the maximum element of the heap to v, v is also non-negative.

You may assume that all the elements, before or after any operations, do not exceed the range of data type `long`. We guarantee that the input sequence is valid, i.e., no deletion while the heap is empty.

Output Every time you read Operation 3, you need to output the current maximum element of the heap in one line.

Sample Input

```
10
1 3
1 5
1 2
3
2
3
4 1
3
1 4
3
```

Sample Output

```
5
3
2
4
```

Problem D - Tree Construction

It is known that, when the pre-order and in-order traversals of a binary tree are given, the structure of this binary tree is determined.

In this problem, you are given the pre-order and in-order traversals of a binary tree. Your task is to output all the leaf nodes in this binary tree.

Input There are several test cases in the input file. The fist line is an integer $t \leq 20$, which indicates the number of test cases.

For each test case, there are three lines. The first line contains an integer $2 \leq n \leq 1000$ representing the total number of nodes in the binary tree. The second line contains n integers, which is the pre-order traversal of this binary tree. The third line contains the corresponding in-order traversal.

You may assume that each node of the tree is labeled by an unique integer between 0 and 10000.

Output For each test case, output all the leaf nodes in one line. The leaf nodes should be ordered from the most left to the most right according to the structure of the binary tree and separated by a space.

Sample Input

```
2
5
7 5 11 3 2
11 5 7 3 2
14
11 12 18 19 20 21 13 14 15 17 16 22 23 24
19 20 18 21 12 14 13 17 15 16 11 22 23 24
```

Sample Output

```
11 2
20 21 14 17 16 24
```

Remark Notice that the binary tree may not be full, i.e., some internal nodes may only have one child.

Problem E - Josephus Problem

Josephus problem is a very famous problem. Given a group of n men arranged in a circle under the edict that every m-th man will be executed going around the circle until only one remains.

The n men in the circle are labeled in order from 1 to n. The n-th man is standing next to the 1-st man. You may assume that the counting starts from 1 and then to 2.

You are asked to output the executed sequence and the survived one at the end.

Input There are several test cases in the input file. The fist line is an integer $t \leq 20$, which indicates the number of test cases.

For each test case, there is only one line containing two integers $1 \leq n \leq 10000$ and $1 \leq m \leq 10000$.

Output For each test case, output the executed sequence and the survived one in one line, separated by a space.

Sample Input

```
2
7 5
5 3
```

Sample Output

```
5 3 2 4 7 1 6
3 1 5 2 4
```

Explanation We consider the first test case. No. 1 starts counting from 1 first. So No. 5 is the first man to be executed. After that No. 6 starts counting from 1 again. No. 1 will count 3 because he is the next man of No. 7. Then No. 3 will count 5 and be executed. Counting will be started from No. 4. Notice that No. 6 will count 2 since No. 5 is already gone. No. 2 will be the next one to count 5 and be executed. Following this rule, No. 6 is the one who survive.

Remark Notice that m can be greater than n.

Problem F - Min Stack

Design a stack that supports `push`, `pop`, `top`, and `get_min`, which is to retrieve the minimum element of the stack.

Input The first line contains an integer $1 \leq n \leq 1000000$, which indicates the number of operations.

The next n lines each contains an operation. There are four different operations in total:

- 1 x: Push an integer x on top of the stack.
- 2: Remove the element on top of the stack.
- 3: Get the top element.
- 4: Retrieve the minimum element of the stack.

You may assume that all the elements are in the range of data type `long`.

Output Every time you read Operation 3, you should output the top element of the stack in one line.

Every time you read Operation 4 you should output the minimum element of the stack in one line.

Sample Input

```
7
1 5
1 3
1 2
3
4
2
4
```

Sample Output

```
2
2
3
```

Source LeetCode Online Judge - Problem 155

Problem G - Finding A Word In A Dictionary

In this problem, you are given a dictionary containing several words. Your task is to answer queries asking whether a single word occurs in the dictionary. We guarantee that all the words are non-empty.

All the words contain only lowercase English alphabets.

Input The first line contains an integer $1 \leq n \leq 1000$, which indicates the number of words in the dictionary. The next n lines each contains a word in the dictionary.

After that, there is one line containing an integer $1 \leq m \leq 100000$, which indicates the number of queries. The following m lines each contains the single word in a query.

The length of any word does not exceed 400.

Output For each query, output one line: "Yes" if the word in that query occurs in the dictionary; "No" otherwise.

Sample Input

```
3
abcde
abcd
cde
4
abcd
abc
cde
bcf
```

Sample Output

```
Yes
No
Yes
No
```

Hint You may consider using a hash table.

Problem H - Two Stacks

This problem asks you to determine a stack-generated permutation in a more general setting.

A permutation of 1 to n is a stack-generated permutation if and only if it can be generated by pushing 1 to n onto a stack and popping them off. For example, stack-generated permutation $(2, 1, 3)$ can be generated by doing operations `push` 1, `push` 2, `pop`, `pop`, `push` 3, `pop`.

In this problem, instead of using only one stack, we use two stacks: every time you can push an element to either stack, or you can pop element from either stack as long as it is non-empty. **However, once an element is popped, it cannot be pushed back to either stack again.**

Your task is to determine whether a permutation can be generated by using two stacks.

Input There are several test cases in the input file. The fist line is an integer $t \leq 20$, which indicates the number of test cases.

For each test case, the first line contains an integer $1 \leq n \leq 1000$. The following line contains n integers separated by a space, which is a permutation of 1 to n.

Output For each test case, output one line: "Yes" if the permutation can be generated by using two stacks; "No", otherwise.

Sample Input

```
4
7
1 3 5 7 2 6 4
9
6 5 4 8 7 9 3 2 1
18
3 4 2 1 5 6 10 8 11 16 18 9 17 7 13 15 12 14
13
4 11 9 12 13 10 5 7 2 6 8 3 1
```

Sample Output

```
Yes
Yes
No
No
```

Explanation We consider the first test case. It can be generated by the following operation sequence: `push` 1 to stack 1, `pop` stack 1, `push` 2 to stack 1, `push` 3 to stack 1, `pop` stack 1, `push` 4 to stack 2, `push` 5 to stack 1, `pop` stack 1, `push` 6 to stack 2, `push` 7 to stack 2, `pop` stack 2, `pop` stack 1, `pop` stack 2, `pop` stack 2.

Remark We recommend you not try this problem until you have finished all previous problems.