Indian Statistical Institute

Semester-I 2018-2019

M.Tech.(CS) - First Year

Lab Test II (20 September, 2018)

Subject: Data and File Structures Laboratory

Total: 60 marks Duration: 4 hrs.

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs18xx-test2-progy.c`
2. When you have finished, copy all your files to `~dfslab/2018/labtest2/cs18xx/`.

1. **(20 marks)** Let us define a *leaky stack* as a special stack data structure from which data items automatically leak over time. Each leaky stack has a certain *leak tolerance* level, L. If the leak tolerance level of a leaky stack is L, then after the completion of every L PUSH or POP operations, the data item at the bottom of the stack automatically leaks out and is lost. Note that, pushing into a full stack (overflow), and popping from an empty stack (underflow) are counted as operations, even though they do not change the stack.

   Write a program to implement the PUSH, POP and DISPLAYSTACK operations on a leaky stack for integer data, given its maximum size (i.e., capacity) and leak tolerance level.

   Input Format

   The first line of input (taken from stdin) is a pair of integers, specifying, respectively, the capacity of the leaky stack, and its leak tolerance level L. The subsequent lines specify a sequence of PUSH and POP operations to be performed on the stack. A push operation is denoted by a leading `+` sign, and a pop operation is denoted by a leading `-` sign.

   Output Format

   Your program should print the contents of the stack on one line after the completion of each PUSH or POP operation, and subsequent leak, if any. The stack contents should be printed even if the operation results in an overflow / underflow. The stack contents should be printed in order from top to bottom as a list of single-space-separated integers.

   Sample Input 0

```
10 3
+ 2
+ 3
+ 5
-
```

   Sample Output 0

```
2
3 2
5 3       ⟵ 2 has leaked out.
3
```

Sample Input 1

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

Sample Output 1

```
1
2 1
3 2        ⟵ 1 has leaked out.
4 3 2
5 4 3 2
5 4 3        ⟵ overflow, followed by a leak.
7 5 4 3
```

Sample Input 2

```
5 2
+ 2
-
-
+ 1
+ 7
```

Sample Output 2

```
2
      ⟵ leak has no effect on an empty stack.
      ⟵ stack is empty.
      ⟵ 1 was pushed, but immediately leaked out.
7
```

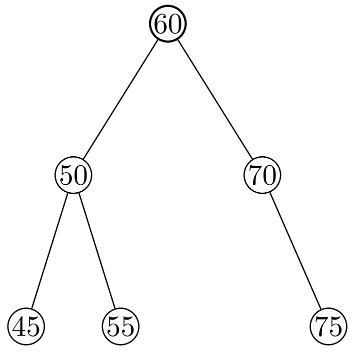
2. **(4 + 8 + 8 = 20 marks)** Write **non-recursive** functions for the *pre-order*, *in-order* and *post-order* traversals of a given binary tree. You may reuse your own implementation of a stack for this program.

   Input Format: Input will be provided via standard input in the following format. The first line of input will contain n, where n is the number of nodes in the binary tree. This will be followed by n more lines. Each of these remaining lines will correspond to one node in the tree and will consist of 3 integers: the data (an integer to be stored in the node), the line number of the node corresponding to the left child (-1 if there is no left child), and the line number corresponding to the right child (-1 if there is no right child).

   Output Format: Your program should print 3 lines, each containing space separated integers, corresponding to the pre-order, in-order and post-order traversals of the given tree.

Sample Input 0:

```
6
60   3   4
50   5   6
70  -1   7
45  -1  -1
55  -1  -1
75  -1  -1
```



Binary Tree for Sample Input 0.

Sample Output 0:

```
60 50 45 55 70 75
45 50 55 60 70 75
45 55 50 75 70 60
```

3. **(20 marks)** Suppose there are N app-cabs in Kolkata, each having a distinct owner-driver. Every day, the i-th owner-driver starts accepting ride requests at b_i, and stops accepting requests at e_i. Write a program to compute the period during the day when maximum cabs are available for taking requests.

   Input Format: Input will be provided via standard input in the following format. The first line of input will contain N, the number of app-cabs / owner-drivers in Kolkata. Each of the following N lines will contain b_i and e_i for $i = 0, 1, 2, \ldots, N-1$. The times will be given in a 24-hour format. For example, $b_1 = 1015, e_1 = 2045$ indicates that driver 1 starts accepting requests at 10:15AM and stops accepting requests at 08:45PM.

   Output Format: Your program should print (in the format described above) the start and end times of the period during which maximum cabs are available, along with the actual number of cabs available during this period. If there are disjoint periods during which the same maximum number of cabs are available, your program should print any one of these periods.

   Sample Input 0:

   ```
   4
   1100 2030
   0600 2100
   1200 2330
   1000 1600
   ```

   Sample Output 0:

   ```
   1200 1600 4
   ```

   Explanation: During 12:00 noon to 04:00PM, the maximum number of cars are available, and this number is 4.

   Sample Input 1:

   ```
   2
   ```

0600 1030
1730 2330

Sample Output 0:
1730 2330 1

Explanation: During 05:30PM to 11:30PM, the maximum number of cars are available, and this number is 1. The following is also a correct answer:
0600 1030 1