INDIAN STATISTICAL INSTITUTE

MTech(CS) I year 2025-2026

Subject: Computing Laboratory

Lab Test 2 (October 14, 2025)

Total: 60 marks Duration: 3 hours

GENERAL INSTRUCTIONS

1. All programs should take the required input via the standard input (terminal/keyboard), and print the desired output to the terminal.
2. Please make sure that your programs adhere strictly to the specified input and output format. Do not print extra strings asking the user for input, debugging messages, etc. These will cause the automatic checking system to fail.
3. Please make sure that the programs are free from memory errors and leaks, you will lose marks if they are not.

Q1. (10 marks) The pseudocode for a **non-recursive** implementation of inorder traversal for a binary tree would look something like the following.

```
1: Create an empty stack S.
2: Push the root node onto S.
3: while S is not empty do
4:     v ← POP(S)
5:     Depending on whether v is a NEW vertex, or has already been SEEN ONCE,
6:     process it appropriately.
7: end while
```

Write a program to implement the above. Your program should print the stack from top to bottom **just before** each POP operation (see line 4). Finally, it should print the indices and values of the nodes in inorder (one node per line). See the sample inputs and outputs provided for details about the format. For your convenience, pictorial representations of the trees in the first few test cases are also provided. In these pictures, the numbers within each node (circle) correspond respectively to the **index** (in small font) and the **value** (in normal font) of the node.

Q2. (10 marks) Given a positive integer N, your task is to print **in sorted order** all integers in $[0, N]$ that are of the form $a^3 + b^3$ (where a and b are integers in $[0, 1000]$, and $a \leq b$). Write a program that accomplishes this task for a fixed $N = 2 \times 10^9$, using the following idea. Build a min-heap, initially containing the N triples

$$(0,\ 0,\ 0) \qquad (1,\ 0,\ 1) \qquad (2,\ 0,\ 8) \qquad \ldots$$

Then, while the min-heap is nonempty, remove the item $(i,\ j,\ i^3 + j^3)$ for which $i^3 + j^3$ is minimum, and print the triple. Then, if $i^3 + (j+1)^3 \leq N$, insert the triple $(i,\ j+1,\ i^3 + (j+1)^3)$ into the heap.

For some a, b, c, d, with $a \neq c$, it may happen that $a^3 + b^3 = c^3 + d^3 \leq N$. These are to be regarded

as **distinct** triples, even though the sum is the same for both triples. Thus, both triples should be printed. If $a < c$, then the triple $(a,\ b,\ a^3 + b^3)$ should be printed before $(c,\ d,\ c^3 + d^3)$.

For this question, there is only one test case, no input files, and one output file.

Q3. (20 marks) You are given two Binary Search Trees (BSTs) T_1 and T_2, in which the nodes store integer values. Your task is to print all the integers stored in T_1 and/or T_2 in sorted order.

Since an inorder traversal of a BST prints the node values in sorted order, the above task may be done using the following simple method: run an inorder traversal on T_1 and store the values in an array, say A_1; repeat the same process on T_2 to obtain a (sorted) array A_2; finally, run the usual linear-time merging algorithm for two sorted arrays on A_1 and A_2. This method requires linear time, and linear (specifically $O(|T_1| + |T_2|)$) extra space.

You need to write a program that accomplishes the task by doing **concurrent inorder traversals** of the two BSTs. The idea is sketched below.

1: Start an inorder traversal of T_1, but stop just before printing the minimum value in T_1.
2: Start an inorder traversal of T_2, but once again, stop just before printing the minimum value in T_2.
3: Compare the values that you were about to print from T_1 and T_2.
4: Print the smaller of the two values, and proceed to the next step of the inorder traversal of the tree that it came from.
5: If both values are the same, print **both** of them, and proceed to the next step of the inorder traversal for both T_1 and T_2.

This method requires $O(h(T_1) + h(T_2))$ extra space. Thus, if the two trees are balanced (we have not yet discussed this in class), you can get by with logarithmic extra space.

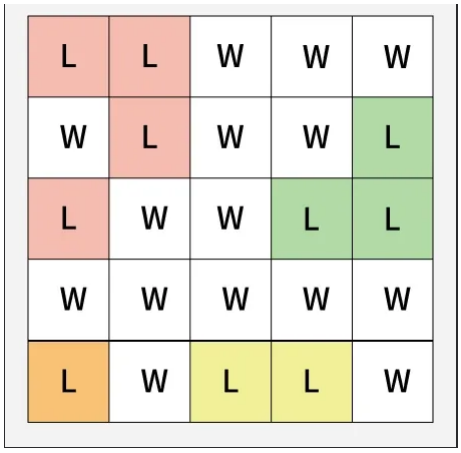
Write a program to implement the above idea. You may reuse any code that you have written for Q1.

Input format: The two BSTs will be given to you via stdin, one after the other, in the usual tabular format that we use in class. Thus, you should be able to call `read_tree(&tree1)` and `read_tree(&tree2)` successively in your program to read the two BSTs.

Output format: A list of all the integers in T_1 and T_2 in sorted order, with one integer per line. If the same integer appears in both BSTs, it should be printed twice.

Q4. (20 marks)

Given an $r \times c$ grid of Water (W) and Land (L), the task is to count the number of islands. An island is a group of adjacent L cells connected horizontally, vertically, or diagonally, and it is surrounded by water or the grid boundary. The goal is to determine how many distinct islands exist in the grid. For the example grid shown below, the output is 4 (the 4 different islands are marked with different colors in the grid). [Hint: Use the Disjoint Set Union (DSU) data structure to efficiently count the number of islands in the grid. After processing the entire grid, iterate through all land cells and use the DSU find operation to determine their root representatives, counting the number of unique sets (i.e., unique roots) to get the total number of islands.]



Input format: The first line of input will contain the values of r and c (comma-separated). This will be followed by r lines, each line specifying a string of length c consisting of L / W. An example input for the grid above would be:

5, 5
LLWWW
WLWWL
LWWLL
WWWWW
LWLLW

Output Format The output should print the number of islands.