Homework 5
CS 601: Advanced Algorithms

Problem 1. [Category: Coding]

Implement a last-in-first-out (LIFO) stack using at most two queues in C++. The implemented stack should support all the functions of a normal stack (push, top, pop, and empty).

Implement the MyStack class:

void push(int x) : Pushes element x to the top of the stack.

int pop(): Removes the element on the top of the stack and returns it.

int top(): Returns the element on the top of the stack.

boolean empty(): Returns true if the stack is empty, false otherwise.

Note that you must use only standard operations of a queue, which means that only push to back, peek/pop from front, size and is empty operations are valid. You can use the C++ built-in Queue data types.

Submit your code file. **[15 points]**

Problem 2. [Category: Coding, Credits to Capstone Exam]

Write a complete C++ program with a Magic function that receives two arrays T_1 and T_2 which represent two binary min-heaps, where T_1 has n nodes and T_2 has m nodes. Each node will be assigned different and distinct integer labels, meaning no two nodes in trees share the same label. The Magic function should generate a new array T_3 that is a binary min-heap constructed out of values present in T_1 and T_2.

Note 1: Your code is permitted to utilize auxiliary space up to O(n+m).

Note 2: Ensure that your code operates with two minimum heaps and constructs a minimum heap from them. No credit will be awarded for code related to maximum heaps.

Note 3: Implement binary trees in your code using basic (static) arrays as the underlying data structure.

Note 4: When executing the program, the user should input distinct and unique label values for the trees.

Note 5: You are not permitted to utilize pre-existing routines that perform the specific tasks we have asked you to do. In other words, using built-in libraries for tasks like array manipulation and heap construction is not allowed. Instead, you are expected to implement these operations from scratch in your code.

What to submit: code file, and with comments in the code file that also gives the time complexity of your program in $O()$ notation. And also in the code file comments, justify how you ensure the auxiliary space complexity of your program is $O(n + m)$. **[15points]**