CSE108 – Computer Programming Lab.

Lab 13

Dynamic memory allocation

Due at 10:00

Hand in: A student with number 20180000001 should hand in a zip file named 20180000001.zip for this lab.

Note: You are required to implement the binary search tree structure as a library with the provided operations listed below.

Part 1. (40 pts) You are given a text file ("input.txt") containing a list of integers separated by spaces. Write a function called **generateBST** that reads the integers from the file and generates a binary search tree (BST) **using pointers, and structures**. Your function should adhere to the following specifications:

```
struct Node {

double Value;

struct Node* Left node;

struct Node* Right node;

} ;


struct BST {

   struct Node* root;

};
```

struct BST* generateBST(const char* filename)

```
{

…

}
```

Your function should return a pointer to the created BST structure. You may assume that the input file exists and is correctly formatted. The integers in the file should be used to build the BST in such a way that each left child is less than its parent, and each right child is greater than its parent. You can use the first integer in the file as root.

Part 2. (30 pts)

Implement the following operations for the previously defined binary search tree structure and give them with a proper menu:

a) **void addNode(struct BST* bst, int value);**

This function should add a new node with the given value to the BST.

b) **void removeNode(struct BST* bst, int value){…}**

This function should remove the node with the given value from the BST, if it exists.

c) struct Node* searchNode(struct BST* bst, int value) {…}

This function should search for a node with the given value in the BST and return a pointer to the node if found, or NULL otherwise.

Part 3. (30 pts)

Implement the following options for printing various statistics of the binary search tree:

a) **int countNodes(struct BST* bst){…}** This function should return the total number of nodes in the BST.

b) **int getMaxDepth(struct BST* bst) {…}** This function should return the maximum depth of the BST (i.e., the length of the longest path from the root to a leaf node).

c) **int countLeafNodes(struct BST* bst) {…}** This function should return the number of leaf nodes (nodes without any children) in the BST.

d) **void printTree(struct BST* bst) {…}** This function should print the binary search tree in an easily understandable format based on the user's choice. You may choose any suitable representation (e.g., in-order, pre-order, post-order, level-order).

General Rules:

1. You will have two hours to provide a solution to the given problem set. You are not permitted to ask any questions. If there is a significant error in the assigned tasks, it will be addressed later.
2. You will be able to hand in your solutions via Teams in the next two hours. The submission will be closed exactly at 10am.
3. There will be an interview session immediately after the submission deadline. Starting at 10am, you will be randomly invited to attend a meeting by a TA to demonstrate your solution and answer any questions asked by the TA.
4. You must be available until 1pm to respond to the demo invitation whenever you receive it. You will have 3 minutes after you are called via Teams. If you do not answer/appear in 3 minutes, you will miss you interview.
5. If you miss your interview or are unable to give satisfactory answers to the questions, you will receive a zero for that lab even if you have submitted your solution.
6. If you have not submitted a solution in time, you will not be invited for the interview and receive zero for that lab.
7. Due to time constraints, some students may not be invited to an interview. In that case, their solutions will be graded offline.
8. Unless you aren't declared for a specific prototype, you may use arbitrary but proper function and variable names that evoke its functionality.

9. The solution must be developed on given version of OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.
10. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it is working partially.
11. Zip your solution file before uploading it to MS Teams. The zip file must contain the C file with your solution and screenshots of the valid outputs of the program.