COMP2113 Programming Technologies / ENGG1340 Computer Programming II

Assignment 4

Deadline: 7 December 2024 (Saturday) 23:59

If you have any questions, please post to the Moodle discussion forum on Assignment 4.

Total marks: 100 marks

General Instructions

Read the instructions in this document carefully.

In this assignment you will solve one simple task and Moodle VPL would automatically test your submitted program. So if your submitted files and program outputs do not conform to our instructions given here, your programs cannot be evaluated and you will risk losing marks totally.

Sample test cases are provided in this document. Note that the test cases may or may not cover all boundary cases for the problem. It is also part of the assessment whether you are able to design proper test cases to verify the correctness of your program. We will also use additional test cases when marking your assignment submission.

Coding environment

Make sure the following compilation command can be used to compile your program when you test our program in Linux environment:

```
gcc -pedantic-errors -std=c11 bst.c –o bst
```

Submission

Name your C program file as **bst.c**. **You need to use VPL to test your code.**

Late submission

If you submit within 2 days after the deadline, there will be 30% mark deduction. If you submit within 3 to 5 days after the deadline, there will be 50% mark deduction. After that, no mark will be given.

Evaluation

Your code will be auto-graded for technical correctness. In principle, we use test cases to benchmark your solution, and you may get zero marks for not being able to pass any of the test cases. Normally partial credits will not be given for incomplete solution, as in many cases the logic of the programs are not complete and an objective assessment could be difficult. However, your work may still be considered on a case-by-case basis during the rebuttal stage.

Academic dishonesty

We will be checking your code against other submissions in the class and from the Internet for logical redundancy. Please be reminded that no matter whether it is providing your work to others, assisting others to copy, or copying others will all be considered as committing plagiarism and we will follow the departmental policy to handle such cases. Please refer to the course information notes for details.

Getting help

You are not alone! If you find yourself stuck on something, post your questions to the course forum, send us emails or come to our support sessions. We want this assignment to be rewarding and instructional, not frustrating and demoralizing. But we don't know when or how to help unless you ask.

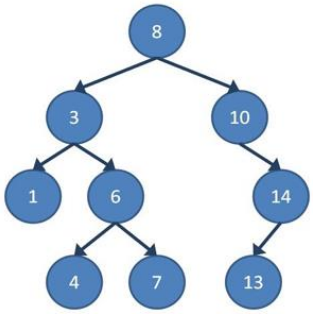
Discussion forum

Please be careful not to post spoilers. Please don't post any code that is directly related to the assignments. However, you are welcome and encouraged to discuss general ideas on the discussion forums. If you have any questions about this assignment you should post them in the discussion forums.

Binary Search Tree in C

A Binary Search Tree has the following characteristics:

- **Binary tree** – Each node in the tree has at most two child nodes (can be one child node or two child nodes).
- **Search tree –** Keys on the left sub-tree must be smaller than the keys on the right sub-tree.
- When searching for a particular key, we do not need to scan through all the items. For example, searching for key 14 in the tree above, we need to do 2 comparisons only.
    - Compare with the root, since 14 is larger than 8, we go to the right path.
    - Compare with node 10, since 14 is larger than 10, we go to the right path.
    - Found key 14.



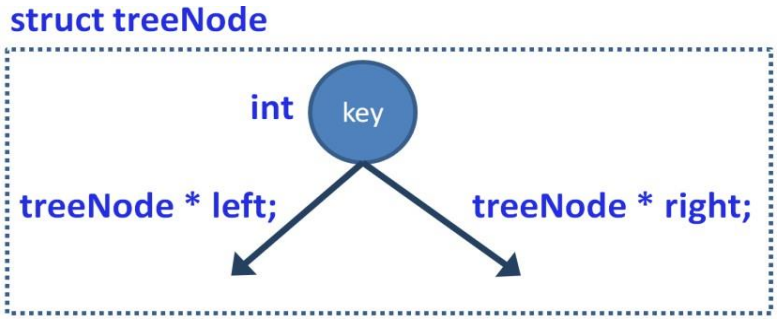
In this problem, you are provided a template program (in VPL). Complete the program so that it can handle the commands below:

- INSERT X
    - Insert the element X into the tree
- PRINT
    - Print all elements in the tree in ascending order
- FINDMIN
    - Look for the minimum element in the tree
- FINDMAX
    - Look for the maximum element in the tree
- FIND X
    - Look for the element X in the tree

You can assume that X is always an integer. Also you can assume that all elements in the tree are unique (that is no two identical elements).

In the template program, a `treeNode` structure is given. It contains three member variables:

- `int key` – the key of `treeNode`
- `struct treeNode *left` – the pointer to the left subtree
- `struct treeNode *right` – the pointer to the right subtree



Besides, the `void Print(treeNode *currentNode)` function has been implemented for you. This function can traverse the tree rooted at `currentNode` from left to right and print the nodes in ascending order of the keys.

You should implement the following functions:

- `treeNode * Insert(treeNode * currentNode, int key);`
  - Insert a new node with key `key` into the tree rooted at `currentNode`. The function should return a pointer pointing the root node.
  - Hint: You should consider at least 2 cases: the tree is empty or the tree is non-empty. Also bear in mind that the key in the left child node must be smaller than the key in the right child node.
- `treeNode * FindMin(treeNode *currentNode);`
  - Look for the minimum key in the tree rooted at `currentNode`. The function should return a pointer pointing the minimum node.
- `treeNode * FindMax(treeNode *currentNode);`
  - Look for the maximum key in the tree rooted at `currentNode`. The function should return a pointer pointing to the maximum node.
- `treeNode * Find(treeNode * currentNode, int key);`
  - Look for the key `key` in the tree rooted at `currentNode`. The function should return a pointer pointing to the target node if found and return NULL if not found.

Hint: All these 4 functions can be implemented in *recursive* manner.

Sample Run

Assume we have `input2_x.txt` in the current directory (available on Moodle). If we run the program like this: (assuming that we are working on Linux)

```
$ gcc -pedantic-errors -std=c11 bst.c -o bst
$ ./bst < input2_x.txt
```

The program will print as follows.

Input file: `input2_1.txt`

```
INSERT 2
INSERT 6
INSERT 4
INSERT 8
INSERT 10
PRINT
END
```

Output:

```
2 4 6 8 10
```

Input file: `input2_2.txt`

```
INSERT 21
INSERT 13
INSERT 40
INSERT 6963
INSERT 2021
PRINT
FINDMIN
FINDMAX
END
```

Output:

```
13 21 40 2021 6963
Minimum element is 13
Maximum element is 6963
```

Input file: `input2_3.txt`

```
INSERT 8
INSERT 10
INSERT 6
INSERT 7
FIND 7
FIND 6
FIND 10
FIND 5
FIND 11
```

```
INSERT 5
FIND 5
MIN
INSERT 0
MAX
INSERT 9999
PRINT
END
```

Output:

```
Element 7 found
Element 6 found
Element 10 found
Element 5 not found
Element 11 not found
Element 5 found
0 5 6 7 8 10 9999
```

Note

- Your program MUST be a C program (NOT a C++ program). Your program will be checked after the submission deadline. Your score will be 0 if we find that your program is not a C program.
- You MUST implement binary search tree (BST) in your program. Your program will be checked after the submission deadline. Your score will be 0 if we find that you use traditional arrays to store keys in your implementation.