COMP2113 / ENGG1340, Assignment 3

Due Date: Dec 5, 2021, 23:59

If you have any questions, please post to the Moodle discussion forum on Assignment 3.

Total marks: 100 marks

- 10 marks for proper code comments and indentation
- 90 marks for program correctness

A maximum of 5 marks will be deducted if you fail to follow the submission instructions strictly.

General Instructions

Read the instructions in this document carefully.

In this assignment you will solve 2 tasks and a tester 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 with each task 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.

Input and output format

Your C / C++ programs should read from the **standard input**. Also, your answer should be printed through the **standard output**. If you failed to follow this guide, the tester may not be able to give a score for your program. Additionally, you should strictly follow the sample output format (including space, line breaker, etc.), otherwise, your answer might be considered as wrong.

How to use the sample test cases

Sample test cases in text file formats are made available for you to check against your work. Here's how you may use the sample test cases. Take Problem 2 test case 3 as an example. The sample input and the expected output are given in the files `input2_3.txt` and `output2_3.txt`, respectively. Suppose that your program is named "2", do the followings at the command prompt of the terminal to check if there is any difference between your output and the expected output.

```
./2 < input2_3.txt > myoutput.txt
diff myoutput.txt output2_3.txt
```

Testing against the sample test cases is important to avoid making formatting mistakes. The additional test cases for grading your work will be of the same formats as the sample test cases.

Coding environment

For Problem 1 on C programming, make sure the following compilation command is used to compile your program:

```
gcc -pedantic-errors -std=c11 1.c –o 1
```

For Problem 2 on C++ programming, make sure the following compilation command is used to compile your program:

```
g++ -pedantic-errors -std=c++11 2.cpp –o 2
```

Submission

Name your programs as the following table shows and put them together into one directory. Make sure that the folder contains only these source files (`*.c, *.cpp`) and no other files. **Compress this directory as a `[uid].zip` file where [uid] is your university number** and check carefully that the correct files have been submitted. We suggest you to download your submitted file from Moodle, extract them, and check for correctness. **You will risk receiving 0 marks for this assignment if you submit incorrect files.** Resubmission after the deadline is not allowed.

Filename	Description
`1.c`	Problem 1
`2.cpp`	Problem 2

Late submission

If submit within 3 days after the deadline, 50% deduction. After that, no mark.

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.

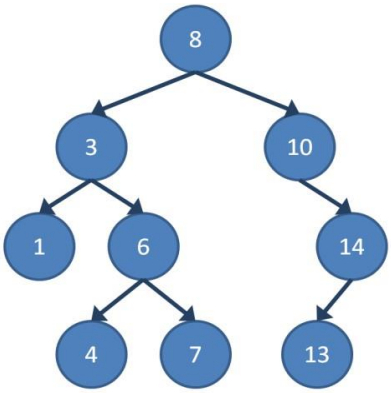
Problem 1: Binary Search Tree in C

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 `1.c`. Complete the program so that it can handle the commands below:

- INSERT X
    - Insert the element X into the tree

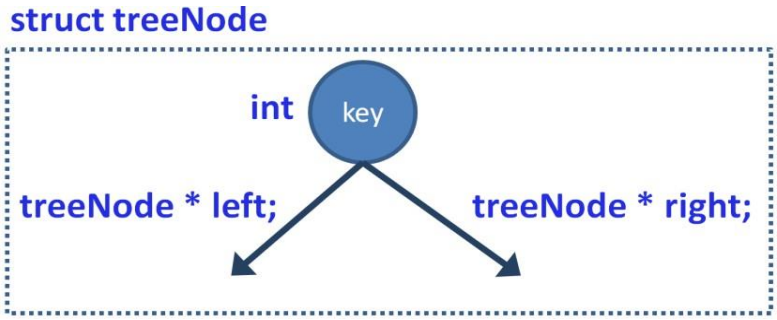


- PRINT
    - Print all elements in the tree in ascending order
- FINDMIN
    - Look for the minimum element in the tree
- FINDMAX
    - Look for the maximum element in the tree
- FIND X
    - Look for the element X in the tree

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 `input1_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 1.c -o 1
$ ./1 < input1_x.txt
```

The program will print as follows.

Input file: `input1_1.txt`

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

Output:

```
2 4 6 8 10
```

Input file: `input1_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: `input1_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.

Problem 2: STL - Log Analyzer

You are provided with a template program `2.cpp`. Complete the program and the program will read and process a web log data file from user input (`cin`). The program will then print out the 5 most popular pages, and the 5 most active users and the corresponding pages they have visited.

The log file consists of records of three types, each record occupies exactly one line. Here is the format of these three types of record:

Page record: `PAGE` *`<page id> <page url>`*
User record: `USER` *`<user id>`*
Visiting record: `VISIT` *`<path id>`*

A **page** record represents a page on the web server. A **user** record represents a user that accesses the system. A **visiting** record represents a visit by the user indicated by the most recent user record. Here is a sample data file:

```
PAGE 1288 /library
PAGE 1282 /home
USER 20686
VISIT 1288
VISIT 1282
USER 20687
VISIT 1288
```

In this case, we have 2 pages (`/library` and `/home`) on the server. Two users have accessed the server, one (#20686) visiting 2 pages (`/library` and `/home`) and the other (#20687) visiting 1 page (`/library`).

You can assume the followings regarding the data file:

- The data file always consists of the three types of records only
- The page ids are unique across all PAGE records
- The user ids are unique across all USER records
- The page ids are unique across all VISIT records of a user
- VISIT records will only appear after the first USER record
- The page id in a VISIT record appears only after its corresponding PAGE record
- There will be at least 5 users and 5 pages

The followings have been implemented for you:

- A `Page` structure, each `Page` object consists of the `id`, `path` and a `counter` to count the number times it is being visited

```
struct Page {
    int id;
    string path;
    int counter;
    Page(int id, string path) {
        this->id = id;
        this->path = path;
        counter = 0;
    };
};

bool operator<(const Page & a, const Page & b) {
    // This function can facilitate sorting
    return (a.id < b.id);
};
```

- A STL vector for organizing the `Page` objects

```
vector<Page> pages;
```

- A `User` structure, each `User` object consists of the `id` and the pages the user visited

```
struct User {
    int id;
    vector<string> visits;
    User(int id) {
        this->id = id;
    };
    void add_visit(int page_id) {
        …
    };
    int size() const {
        return visits.size();
    };
    void print_visits() {
        …
    }
};
```

- A STL vector for organizing the User objects

```
vector<User> users;
```

- A function to overload < operator for the comparison of 2 pages based on their ids

```
bool operator<(const Page & a, const Page & b) {
    return (a.id < b.id);
};
```

You are required to complete the missing functions in the template program to it work.

Note: The functions `Page()` and `User()` in the structs `Page` and `User` above are called **struct constructors**. Refer to Module 9 "C Programming (Part 3)" for the discussion on the struct constructor functions. It works the same in C++.

Sample Run

Assume we have `input2_1.txt` in the current directory (available on Moodle).
If we run the program like this: (assuming that we are working on Linux)

```
$ g++ -pedantic-errors -std=c++11 2.cpp -o 2
$ ./2 < input2_1.txt
```

The program will print:

```
*** 5 most popular pages ***
36:/msdownload
32:/ie
17:/products
17:/search
13:/sitebuilder
*** 5 most active users ***
23:10068
- /activeplatform
- /activex
- /automap
- /frontpage
- /gallery
- /ie
- /intdev
- /isapi
- /java
- /msdownload
- /musicproducer
- /office
- /promo
- /sbnmember
```

```
- /search
- /sitebuilder
- /spain
- /vbasic
...
[snipped]
...
11:10019
- /athome
- /clipgallerylive
- /games
- /isapi
- /kb
- /logostore
- /msdownload
- /msoffice
- /ntserver
- /products
- /search
```

Note

- If two pages are equally popular, the pages are ordered lexicographically.
- If two users visit the same number of pages, the users are ordered by their id ascendingly.
- The list of pages a user visit must be printed in ascending order.
- You MUST use STL vector 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 pages and users in your implementation.