ENGG1340 Computer Programming II / COMP2113 Programming Technologies

Assignment 3

Deadline: 28 April (Friday) 23:59

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

Total marks: 100 marks

General Instructions

Read the instructions in this document carefully.

For this assignment, you are required to complete two tasks, and a tester will automatically evaluate your submitted program. If your submitted files and program outputs do not comply with the instructions provided, your programs will not be evaluated, and you may risk losing marks entirely.

Sample test cases are included with each task in this document. It is important to note that the test cases may or may not cover all boundary cases for the problem. It is also part of the assessment to determine whether you can design appropriate test cases to verify the correctness of your program. Additional test cases will be used to mark your assignment submission.

Coding environment

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

```
g++ -pedantic-errors -std=c++11 1.cpp -o 1
```

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

```
gcc -pedantic-errors -std=c11 2.c -o 2
```

Submission

Please name your programs as shown in the table below and place them together in a single directory. Ensure that the folder contains only the source files (1.cpp, 2.c) and no other files. **Compress the directory as a [uid].zip file, where [uid] represents your university number**, and double-check to ensure that the correct files have been submitted. We recommend downloading the submitted file from Moodle, extracting it, and verifying its accuracy. **If you submit incorrect files, you may risk receiving a 0 for this assignment**, and resubmission after the deadline is not allowed.

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

Late submission

If you submit it within 3 days after the deadline, there will be a 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 check 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: C++ STL [50%]

You are required to develop an application to maintain student exam grades. This application allows you to enter, search information of students, and view all exam marks of the students based on different rules. The primary purpose of building this application is to store and view student examination records. A student structure is defined as follows:

```
struct Student {
      int id;
      string name;
      Student() {
            id = 0;
            name = "invalid";
      };
      Student(int inputid, string inputname) {
            id = inputid;
            name = inputname;
      };
};
```

A table class is defined as follows. Note that **class** is optional in our course. You are not required to work on the class but just implement some member functions inside.

```
class Table {
public:
     void InsertStudent(Student x, int y);
     void SearchbyID(int x);
     void SearchbyGrade(int y);
     void Statistics();
     void PrintAll();
     // You can add more functions if you want.
private:
     /* The data type is decided by you */ records;
};
```

We would like to implement a program *1.cpp* that supports five commands, namely InsertStudent, PrintAll, SearchbyID, SearchbyGrade, and Statistics.

The program source code is shown below.

```
#include<iostream>
#include<string>
#include<algorithm>
#include<map>
#include<vector>
// You can add more libraries here (if needed)

using namespace std;

// Define structure, classes, and member functions
struct Student {
     // Code given above
};

class Table {
     // Code given above
};

bool operator<(const Student&a, const Student&b) {
     // This function may be required by your implementation.
}

//insert one record into the stored records
void Table::InsertStudent(Student x, int y) {
     // To be implemented
}

//return the name and grade of the student with id x
void Table::SearchbyID(int x) {
     // To be implemented
}

//return the id and name of the student with grade y
void Table::SearchbyGrade(int y) {
     // To be implemented
}

void Table::Statistics() {
     // To be implemented
}

//Print all records in the ascending order of id
void Table::PrintAll() {
     // To be implemented
}

int main() {
     Table t;
     string command;
     int id;
```

```
    string name;
    int grade;

    while (cin >> command) {
        if (command == "InsertStudent") {
            cin >> id >> name >> grade;
            Student s = {id, name};
            t.InsertStudent(s, grade);
        }
        else if (command == "SearchbyID") {
            cin >> id;
            t.SearchbyID(id);
        }
        else if (command == "SearchbyGrade") {
            cin >> grade;
            t.SearchbyGrade(grade);
        }
        else if (command == "PrintAll") {
            t.PrintAll();
        }
        else if (command == "Statistics") {
            t.Statistics();
        }
        else if (command == "exit") {
            break;
        }
    }
    return 0;
}
```

1. InserStudent *[id] [name] [grade]* - Insert the record with ID, name, and grade into the table. For example, we issue the following commands to insert 6 records into the table.

```
InsertStudent 2022001 Chim 100
InsertStudent 2022002 Qian 97
InsertStudent 2022003 Jolly 88
InsertStudent 2022004 Kevin 81
InsertStudent 2022005 Kiu 97
InsertStudent 2022006 Tim 79
```

- You can assume that the name of the students will not contain any space
- If a student with the ID is already in the graph, output “Student exists.”.
- The ID of the students may not start from 0 and may not be in ascending order.

- Decide a suitable STL container to implement the Table::records member variable. We need to access to a student and grade given the student ID in other operations. Some container may greatly simplify your program ☺.

2. PrintAll - Before you proceed to the next part, you may implement PrintAll() to output the name of all records in the table object in the ascending order of the student ID and validate the correctness of your program.

```
PrintAll
2022001 Chim 100
2022002 Qian 97
2022003 Jolly 88
2022004 Kevin 81
2022005 Kiu 97
2022006 Tim 79
```

3. SearchbyID *[id]* - Search for a student with the ID in the table.
   For example, we issue the following command to search for the name and grade in the table.

```
SearchbyID 2022003
Jolly
88
```

   - If there's no record in the table with student ID equal to the input, output “No such student” on screen

4. SearchbyGrade *[grade]* - Print the student IDs and names with this grade.

```
SearchbyGrade 97
2022002 Qian
2022005 Kiu
```

   - If there is no student with this grade, output “No such record.”.
   - If there are more than one records, output them in ascending order of the student ID, line by line.

5. Statistics - Print the maximum, minimum, and median of grades

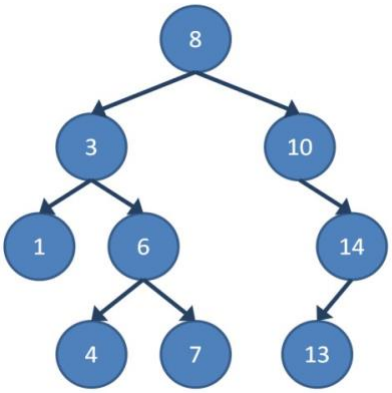
```
Statistics
Maximum 100
Median 92.5
Minimum 79
```

- The median is the middle of a sorted list of grades.
- If there are even amount of records, you must calculate the average of the middle pair.

Problem 2: Binary Search Tree in C [50%]

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 only need to make 2 comparisons.
    - 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 2.c. Complete the program so that it can handle the commands below:

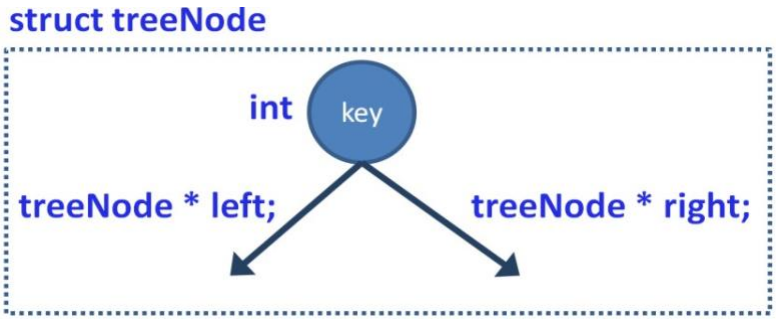
- INSERT X
    - Insert the element X into the tree
- PRINT

    - 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 to the root node.
    - Hint: You should consider at least 2 cases: the tree is empty, or the tree is non-empty. Also, remember that the key in the left child node must be smaller than the key in the right.
- `treeNode * FindMin(treeNode *currentNode);`
    - Look for the minimum key in the tree rooted at `currentNode`. The function should return a pointer pointing to 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 recursively.

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 2.c -o 2
$ ./2 < 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
FINDMIN
INSERT 0
FINDMAX
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
Minimum element is 5
Maximum element is 10
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 a 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.

If you face any problems, **please feel free to contact us**! **We are very happy to help you!**
We wish you enjoy this assignment! ☺