COMP2123 Programming technologies and tools

Assignment 3 – Use of STL

Deadline: Dec 13, 23:00

Dear Students,
Hello! If you have any questions w.r.t. this assignment, please first visit our Assignment 3 Discussion Forum to see if you can find our answer there. If not, please post your questions in the Forum. If you would like to ask through email, please feel free to send email to Max (jianyu@hku.hk), and cc email to Dr. Cui (heming@cs.hku.hk). We are very happy to answer your questions.

- For all questions, please submit them through Moodle, and click the submit button before the deadline.
- Please double check your submission, check ALL of these to ensure that your assignment is submitted properly:
  - Check your email for acknowledgement from Moodle, keep it for record.
  - Check the assignment page again and check if the status is changed to "Submitted for grading".
  - Check the assignment page to see all files you have submitted. Please contact us as soon as possible if you encounter any problem regarding the submission. Enjoy this assignment and let us know if you need our help.

TA Max and Dr. Cui

Please submit your answers in Moodle.

1. [60%] Suppose a school has a system that manages students' profiles (personal information) and their grades in different classes. The system has following functions:
   1) Create or delete students' profile.

      A student profile consists of studentID (string), name (string) and phoneNumber (string) of a student.

   2) Create or delete a grade record.

      A grade record consists of studentID (string), classID (string) and a mark (double), capturing the mark that the student get in a particular class.

   3) Given a classID, display the marks of the students in that class (the definition of the display function will be illustrated through examples in the later parts.)
   4) Display the profiles of all students in the school (the definition of display function will be illustrated through examples in the later parts.)

   You may define two structs to model students' profiles and grade records:

```
/* database.h */
typedef struct profile_t {
        //Define variable to store a student profile
} Profile;
typedef struct grade_t {
        //Define variables to store the grade for a student in a specific class
} Grade;
```

You may define a class Database to store all students' profiles and grade records.

```
/* database.h */
class Database {
        private:
          // Define some data structure using STL containers here to store all students' profiles and
        the grade records.

        public:
          bool add_profile(string id, string name, string phone);
          bool del_profile(string id);
          bool add_grade(string id, string class_id, double grade);
          bool del_grade(string id, string class_id);

        //Display data
          void display_profiles();
          void display_grades(string class_id);

};
```

We would like you to define the necessary member variables for struct Grade and Profile first, and then define some data structures for class Database to store all students' profiles and grade records. Finally, you need to implement 6 functions in class Database, namely add_profile, del_profile, add_grade, del_grade, display_profile and display_grades.

The source code of the program is showed below.

```
/* database_main.cpp */
/* header files definition, omitted in this document */
int main() {
  string command, command_type;
  string id, name, phone, class_id;
  double grade;
  Database database;
  while(cin >> command >> command_type) {
    if(command_type == "profile") {
      if (command == "create") {
        cin >> id >> name >> phone;
        database.add_profile(id, name, phone);
      } else if (command == "delete") {
        cin >> id;
        database.del_profile(id);
      } else if (command == "display") {
        database.display_profiles();
      }
    } else if (command_type == "grade") {
      if (command == "create") {
```

```
      cin >> id >> class_id >> grade;
      database.add_grade(id, class_id, grade);
    } else if (command == "delete") {
      cin >> id >> class_id;
      database.del_grade(id, class_id);
    } else if (command == "display") {
      cin >> class_id;
      database.display_grades(class_id);
    }
   }
  }
}
```

The 6 functions are defined as follow:

1) add_profile(x, y, z) – add a student profile with x as student ID, y as student name and z as phone number. For example, you may issue the following commands to add 2 student profiles.

```
create profile 303000000 John 11223344
create profile 303000001 Ted 11223355
```

   - The function outputs nothing when a new profile record is created, or update the profile and output **Profile Updated** when the Record Exists (a record with the same studentID exists).
   - The function always returns true.

2) del_profile(x) – delete a student profile with x as student ID. For example, you may issue the following commands to delete 2 student profiles.

```
delete profile 303000000 //delete John
delete profile 303000002 //show Record Not Exists as corresponding profile is not in system
```

   - The function outputs nothing when succeed or **Record Not Exists** when fail.
   - The function returns true when succeed or false otherwise.
   - You also need to delete the corresponding records of grade when deleting a student profile.

3) add_grade(x, y, z) – add a grade record with x as student ID, y as class ID and z as the corresponding grade. For example, you may issue the following commands to delete three grade records.

```
create grade 303000000 comp2123 90 //Show nothing
create grade 303000003 comp1117 80 //Output Student Not Exists
create grade 303000000 comp2123 98 //Update John's grade in comp2123
```

   - The function show **nothing** when succeed, or update the grade record and output **Grade Updated** when the Record Exists (a record with the same studentID and classID exists), and outputs **Student Not Exists** when student with such student ID does not exists in the system.
   - Two grade record is identical if they have the same studentID and classID.
   - The function return true when succeed and false otherwise.

4) del_grade(x, y) – delete a grade record with x as student ID and y as class ID. For example, you may issue the following commands to delete 2 grade record.

```
delete grade 303000000 comp2123 // delete John's grade in class comp2123 in the system
delete grade 303000000 comp2222 // show Record Not Exists
```

- The function output nothing when succeed or **Record Not Exists** when fail.
- The function return true when succeed or false when fail.

5) display_profiles – show all student profiles. For example, you may issue the following commands to show all student profiles.

```
display profile
// Student profiles are showed below
Amy 30XXXXXX1 12345678
Batman 30XXXXXX2 23456781
Cat 30XXXXXX3 34567812
```

- The function output all student profiles in the system.
- Student profiles are display in an ascending order by their **name**.

6) display_grade(x) – show all grade records with class ID being x. The maximum, minimum and average grades will also be showed. For example, you may issue the following commands to show all student grade records in class comp2123.

```
display grade comp2123
//Student grades are showed below
Batman 30XXXXXX1 70.0
Amy 30XXXXXX2 80.0
Cat 30XXXXXX3 90.0
Min 70.0
Max 90.0
Average 80.0
```

- The function output all grade records (along with maximum, minimum and average grade) with corresponding class ID when there exists at least one record, and output **No Records Found** otherwise.
- The grade record is displayed in an ascending order by their **mark**.

Your tasks:

1) Finish defining struct Grade, struct Profile and class Database in database.h. Then, you should implement the class Database. You do not need to modify database_main.cpp.
2) submit all three files database.h, database.cpp and database_main.cpp(unmodified) to Assignment 3 Question 1 VPL. We provide some test cases (not the marking test cases) in the VPL and you can evaluate your program there.

2. [40%] Consider S is a sequence of integers, and it is defined as follow:

$$S =< S_1, S_2, \dots, S_n >$$

For example, $S_x =< 1,5,2,3,1 >$ is a sequence with 5 integers.

We define the length of a sequence, denoted as len(S), as the number of integers in sequence S. For example, $len(S_x) = 5$.

We further define $D(S)$ as follows:

$$D(S) = s_1^2 + s_2^2 + \cdots + s_n^2$$

For example, $D(S_x) = 1^2 + 5^2 + 2^2 + 3^2 + 1^2 = 40$

Consider two sequences S_a and S_b, we have the following definition:

$$\begin{cases} if\ len(S_a) < len(S_b), then\ S_a < S_b \\ if\ len(S_a) = len(S_b), then \begin{cases} if\ D(S_a) < D(S_b), \text{then } S_a < S_b \\ if\ D(S_a) = D(S_b), \text{then } S_a = S_b \\ if\ D(S_a) > D(S_b), then\ S_a > S_b \end{cases} \\ if\ len(S_a) > len(S_b), then\ S_a > S_b \end{cases}$$

Suppose there is a program that takes n sequences as input, and outputs all the sequence in an ascending order according to the definition above.

The input/output are given as follow.

Input:
N(total number of sequences)
M1(length of sequence) S11 S12 S13 …
M2(length of sequence) S21 S22 S23 …
...
(there are N lines of sequences, each line takes length of the sequence as input first, and then input integer of the sequence one by one, separated by space)
Output:
Sequence S_{min}
…
Sequence S_{max}

You may define a class Sequence to represent a sequence of integer.

```
/* sequence.h */
class Sequence {
private:
  //Define some variable to store this sequence and its length
public:
  Sequence(){}
};
```

A function sort_list is defined to sort multiple sequences, after calling this function, seq should have an ascending order of Sequences.

```
/* sequence.h */
void sort(vector<Sequence> &seq);
```

You may also need to overload c++ output function for vector<Sequence> and input function for Sequence, to support usage like cout << sequence_list and cin >> tmp_seq in sequence_main.cpp.

The source code of the program is given as follow. In this program, multiple Sequences are stored in vector<Sequence>.

```
#include <iostream>
#include <vector>
#include "sequence.h"
using namespace std;
int main() {
 vector<Sequence> sequence_list;
 // n is number of sequence
 int n;
 cin >> n;
 for (int i = 0;i < n;i++){
  Sequence tmp_seq;
  //get a sequence from standard input
  cin >> tmp_seq;
  sequence_list.push_back(tmp_seq);
 }
  //sort multiple sequences
  sort_list(sequence_list);
  cout << sequence_list;
}
```

Finally, you may have a program with the following input/output.

```
// When input is the following (4 sequences, and their length are 3, 1, 3 and 3 individually).
                    4
                    3 1 2 3
                    1 5
                    3 5 6 7
                    3 9 1 3
// The output of this program would be like this
                    5
                    1 2 3
                    9 1 3
                    5 6 7
```

Your tasks:

1) Finish defining class Sequence and implement class Sequence, function sort_list and overlapping functions mentioned above along with other necessary functions in sequence.h and sequence.cpp, to finalize the program.
2) You do not need to modify sequence_main.cpp
3) Submit all three files: sequence.h, sequence.cpp and sequence_main.cpp(unmodified) to Assignment 3 Question 2 VPL. We provide some test cases (not the marking test cases) in the VPL and you can evaluate your program there.