Creative Software Programming Assignment#3 (week-3)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

In this week **Handed out will be Sep 17, 2020, Due Sep 22, 2020**

1. Contacts Manager

We want to develop the address book we created last assignment(week-2, contacts).

Refer to the given skeleton code and complete the program to work correctly.

The assignment directory structure should be follow:

- week-3
  - contacts
    - contacts.cc // Code you need to complete
    - contacts.h // Definition of contacts type
    - student.h   // Definition of student type
    - utility.h   // memory allocator

You only need to edit the `contacts.cc` file.

Find the commented `TODO:` flag in the code to complete the function.
The code you implement should satisfy the following comment of each function.

The function signature to be implemented is as follows.

```
bool add(student_t student);
bool add(int id, std::string name);
bool add(int id, std::string name, std::string phone);
bool add(int id, std::string name, std::string phone, std::string mail);

bool remove(int id);

bool update(int id, std::string name, std::string phone, std::string mail);

student_t find(int id);

size_t size();
```

Helps

You haven't learned it, but you might think that the variables you need to use are already declared. Think of the `students` and `m_size` variables defined globally anywhere in the function you need to implement.

In remove operation, you can explicitly express that it has removed by marking the student id as an unusable value such as -1.

Guide for beginners

Our purpose is to fill in the contents of the following 8 functions.

```
bool add(student_t student);
bool add(int id, std::string name);
bool add(int id, std::string name, std::string phone);
bool add(int id, std::string name, std::string phone, std::string mail);

bool remove(int id);

bool update(int id, std::string name, std::string phone, std::string mail);

student_t find(int id);

size_t size();
```

Add

First, let's look at the add function. The process of saving the student with the information entered must be carried out.

We already have an array of `studet_t` named `students` declared to be of a certain size.
We will find a space that has not already been assigned a student and add the students that have been entered.

First, how do you find an empty space?, consider a loop through students.

```
for (int i = 0; i < m_size; i++) {
    // students[i]
}
```

We have to check two things as we tour the students.
First of all, duplicate students cannot enter, so the ID of the entered student and the existing ID must not match either. And the input student will be added to the first space marked with an empty space such as -1 with id.

To put it simply, we can think of the following two loops. The rest of the add functions can work this way, or you can use function overloading!

It would of course be possible to reduce the loop to one by using an additional variable.

```
bool contacts_t::add(student_t student) {
    // check duplicate
    for (int i = 0; i < m_size; i++) {
        if (students[i].id == student.id) {
            return false;
        }
    }

    // find empty space and put student
    for (int i = 0; i < m_size; i++) {
        if (students[i].id == -1) {
            students[i] = student;
            return true;
        }
    }
```

```
    return false;
}
```

Remove

Now let's see how to write remove function.

First you just need to find the student that matches the given id and mark its id as -1.

(*The -1 is for convenience. You can set a specific value that will not be entered as an id, or you can add a variable indicating that it has been deleted.*)

So, it loops through the `students` and searches if there is a student with the same ID and if so, deletes it and returns `true`.

```
bool contacts_t::remove(int id) {
    for (size_t i = 0; i < m_size; i++) {
        if (students[i].id == id) {
            students[i].id = -1;
            return true;
        }
    }

    return false;
}
```

Remain parts

The configuration of the rest of the function is similar. You can traverse the `students` to find students with a given id, update information (*update function*), or return the student itself(*find function*). If you make the process of finding the id itself as a separate function, you can increase code reusability.

```

```