DigiPen
INSTITUTE OF TECHNOLOGY

Programming Evaluation

Module 2 – High-level Programming II

Purpose

This assignment gives you practice most topics covered this semester, classes, overloaded operators, pointers, dynamic memory allocation/deallocation, STL container and file input output.

Information

The goal is to implement a class called PhoneBook which allows the users to store phone entry information. A phone entry contains the name, last name, email, region and number of a contact. There are several methods that manipulate the phone book and its entries. These methods include loading/saving a phonebook from a file, adding/removing entries and sorting etc… This is the interface to the class:

```
struct PhoneEntry
{
    std::string name;
    std::string last_name;
    std::string email;
    int region;
    int number;

    PhoneEntry(std::string name_, std::string last_name_,
               std::string email_, int region_, int number_);
};

class PhoneBook
{
public:
    PhoneBook(std::string filename_);
    PhoneBook(const PhoneBook &copy_);
    PhoneBook & operator= (const PhoneBook & rhs_);

    void Save(std::string filename_) const;
    friend std::ostream & operator<<(std::ostream & os_, const PhoneBook & phonebook_);
    friend std::istream & operator>>(std::istream & os_, const PhoneBook & phonebook_);

    void PhoneBook::AddEntry(const std::string &name_,
                             const std::string &last_name_,
                             const std::string &email_,
                             int region_, int number_);

    void SortByName();
    void RemoveEntriesByName(std::string subString_);

    ~PhoneBook();

private:
    std::vector<PhoneEntry *> entries;
    //std::deque<PhoneEntry *> entries;
};

std::ostream & operator<<(std::ostream & os_, const PhoneBook & phonebook_);
std::istream & operator>>(std::istream & os_, const PhoneBook & phonebook_);
```

Note: The contact information inside the file are stored as follows:

```
John Doe
JDoe@hotmail.com
206-4958374
*
```

which represents the following:

```
Name LastName
Region-Number
Email Address
*(for separation between contacts)
```

Method	Description
`Non default Constructor`	Given a file name, this method will open the file and load all the entries found in the file. The file should use the input operator in order to load data from a file.
`Copy Constructor`	Since entries are dynamically allocated, a copy constructor must be implemented.
`operator=`	Since entries are dynamically allocated, an assignment operator must be implemented.
`Operator<<`	The output operator is responsible for putting the content of a phonebook in any output stream (can be usedwith ***std::cout*** or an open output file).
`Operator>>`	The input operator is responsible for reading the content of a phonebook from any input stream (can be usedwith ***std::cin*** or an open input file).
`Save`	Given a file name, this method will open the file and save all the entries found in the phonebook into the file. The file should use the ouput operator in order to save the data into the file.
`AddEntry`	Given a name, last name, email, region and a number, this method will dynamically allocate a phone entry and push it into the phonebook's container.
`SortByName`	This method sorts the content of a phonebook by first name. **Note:** • Use the sort method found in the STL container's library in order to sort the entries. • You will need to create a global function that teaches the sort function how to sort the data found in the container. • Very important, the data in the container is of type ***PhoneEntry ** and NOT ***PhoneEntry***.

`RemoveEntriesByName`	Given a substring, this method will remove all entries that contain the substring in its name. ***Note:*** ***Make sure you properly remove elements from the container (no crashes, no memory leak).***
`Destructor`	Removes all elements from the container and makes sure all memory is deleted properly.

NOTE:

- ***Your code will be tested with two types of containers (vector and deque).***
- ***You are allowed to add private helper functions in your classes and one global function that teaches the sort algorithm how to sort your container.***

Testing your code

A "Grading Scripts" folder is provided in order for you to check if your implementation is correct.

In order to run the tests follow these steps:

- grab both the cpp file ***"PhoneBook.cpp"*** and the header file ***"PhoneBook.h"*** and place them in the "**Grading Scripts**" folder.
- Double click on the **"RunAll.cmd"** file.
    - All tests will run automatically. For every test, you will know if you passed it or failed it. If you failed the test, the scripts will tell you why you failed it.

      NOTE: ***If you are working on a DigiPen lab/classroom machine you need to run the "DigiPen-RunAll.cmd" file instead of "RunAll.cmd"***

What to submit

You must submit bot ***PhoneBook.cpp*** and ***PhoneBook.h*** files in a single .zip file (go to the class page on moodle and you will find the assignment submit link).

Do not submit any other files than the ones listed.