Creative Software Design, Assignment 5-2

Deadline: 2024-10-08 23:59 (No score for late submission)

- Submit your homework by uploading your zip file to the LMS assignment section. Below is an example.

```
13178_Assignment1-1_2024123456.zip
├─ 1.cc
├─ 2.cc
├─ 3.cc
└─ ...
```

- Your zip file name should follow this format: **13178_Assignment[Assignment-number]_[Student-ID].zip**
  - Ex. 13178_Assignment1-1_2024123456.zip

- Source files should be named as **<filename>.cc** *or* **<filename>.cpp**

- **You must submit your solution in the zip file before the deadline.**

1. Write a C++ program for a sorted number array.

   A. **Implement the `SortedArray` Class:**

      1. **Private members:**

         i. `std::vector<int> numbers_`: Stores the list of numbers.

      2. **Public members:**

         i. `SortedArray()`: the constructor.

         ii. `~SortedArray()`: the destructor.

         iii. `void AddNumber(int num)`: add the `num` into the `numbers_`.

         iv. `std::vector<int> GetSortedAscending()`: returns the `numbers_` in ascending order.

         v. `std::vector<int> GetSortedDescending()`: returns the `numbers_` in descending order.

         vi. `int GetMax()`: returns the maximum number in the `numbers_`.

         vii. `int GetMin()`: returns the minimum number in the `numbers_`.

   B. `main()` **Function:**

      1. Continuously prompts the user for commands.

      2. Processes commands or quit the program.

   C. **Requirements:**

      1. Use **STL functions** such as `std::sort`, `std::max_element`, and `std::min_element` for sorting and min/max operations.

      2. **DO NOT** implement your own sorting and min/max code.

      3. The program takes commands repeatedly until the user enters `quit`.

D. Example output of your program (Bold text indicates user input):

```
9 3 6 2 7⏎
ascend⏎
2 3 6 7 9
decend⏎
9 7 6 3 2
max⏎
9
min⏎
2
10 3⏎
ascend⏎
2 3 3 6 7 9 10
quit⏎
```

E. Submission file: one C++ source file (File name: 1.**cc** or 1.**cpp**)

2. Write a C++ program for an answering machine.

A. Input:

1. `add {phone number} {message string}:`

   Save a message for the given phone number. If a message already exists, overwrite it. Use `std::getline()` to handle spaces in the message string.

2. `delete {phone number}:`

   Delete the message for the given phone number.

3. `print {phone number}:`

   Print the message for the given phone number. If no message exists, print an empty string.

4. `list:`

   List all phone numbers and their corresponding messages.

5. `quit:`

   Quit the program.

B. Output:

1. The program should display the result of each command.

C. **Implement the `MessageBook` class in the provided skeleton:**

```
#include <map>
#include <string>
#include <vector>

using namespace std;

class MessageBook {
public:
   MessageBook();
   ~MessageBook();

   void AddMessage(int number, const string& message);
   void DeleteMessage(int number);
   vector<int> GetNumbers();
   const string& GetMessage(int number);

private:
   map<int, string> messages_;
};
```

D. Requirements:

1. The program takes commands as described, repeatedly until the user enters `quit`.
2. DO NOT add more functions in the `MessageBook` class.
3. All commands should be processed in the `main()` function.

E. Example output of your program (Bold text indicates user input):

```
add 1112222 hello⏎
add 2231144 nice to meet you⏎
add 1234321 too⏎
print 2231144⏎
nice to meet you

list⏎
1112222: hello
1234321: too
2231144: nice to meet you
delete 1112222⏎
list⏎
1234321: too
2231144: nice to meet you
quit⏎
```

F. Submission file: one C++ source file (File name: 2.**cc** or 2.**cpp**)

3. Write a C++ program for integer set operations.

A. Input:

1. `{ num`$_{j1}$` num`$_{j2}$` ... num`$_{jn}$` } OP { num`$_{k1}$` num`$_{k2}$` ... num`$_{kn}$` }`

2. Operations (`OP`):

   i. `+` : Union

   ii. `*` : Intersection

   iii. `-` : Difference

3. `0` : Quit the program.

B. Output:

1. Display the resultant set of the operations.

C. Implement the functions in the provided skeleton:

```
#include <set>
#include <string>

using namespace std;

set<int> parseSet(const string& str);
void printSet(const set<int>& set_);
set<int> getUnion(const set<int>& set0, const set<int>& set1);
set<int> getIntersection(const set<int>& set0, const set<int>& set1);
set<int> getDifference(const set<int>& set0, const set<int>& set1);
```

D. Requirements:

1. Use STL's `std::set` for storing and manipulating integer sets.

2. The program takes commands as described, repeatedly until the user enters `0`.

3. All commands should be processed in the `main()` function.

E. Example output of your program (Bold text indicates user input):

```
{ 1 2 3 } + { 3 4 5 }⏎
{ 1 2 3 4 5 }
{ -1 5 3 2 } - { 1 2 3 }⏎
{ -1 5 }
{ -1 5 3 2 } * { 1 2 3 }⏎
{ 2 3 }
0⏎
```

F. Submission file: one C++ source file (File name: 3.**cc** or 3.**cpp**)