Creative Software Programming, Assignment 7-2

Handed out: Oct 21, 2021

Due: 23:59, Nov 2, 2021 **(NO SCORE for late submissions!)**

- *Only files submitted by **git push to this course project at** https://hconnect.hanyang.ac.kr (<Year>_<Course no.>_<Class code>/<Year>_<Course no.>_<Student ID>.git) will be scored.*
- *Place your files under the directory structure **<Assignment name>/<Problem no.>/<your files>** just like the following example.*
  - ***DO NOT push CMake intermediate output files** (CMakeCache.txt, cmake_install.cmake, Makefile, CMakeFiles/*).*
  - ***DO NOT use cmake_minimum_required()** command in CMakeLists.txt.*

```
+ 2020_ITE0000_2019000001
  + 2-1/
    + 1/
      - 1.cpp
      - CMakeLists.txt
    + 2/
      - 2.cpp
      - CMakeLists.txt
    + …
```

- *The submission time is determined not when the commit is made **but when the git push is made**.*
- *Your files must be committed to the **master branch**. Otherwise, it will not be scored.*
- *Your program should output correct results even for inputs other than those used in the example.*
- *Basically, assignments are scored based on the output results. If it is not possible to check whether a requirement is implemented because the output is not correct, no score is given for the requirement, even if it is implemented internally. However, even if the output result is correct, no score is given for a requirement if the internal implementation does not satisfy the requirement.*

1. Write a program for an answering machine
    A. Implement class MessageBook in the code skeleton.
    B. This program should take user input repeatedly
    C. Input:
        i. 'add' [phone number] [message string] – Save [message string] for [phone number]
            1. If you save a new message to the number that already has a message, the previous message is overwritten.
            2. [message string] should be able to contain spaces. You may need to use std::getline().
        ii. 'delete' [phone number] – Delete the saved message for [phone number]
        iii. 'print' [phone number] – Print out the saved message for [phone number]. If there is no message for [phone number], just print out an empty string.
        iv. 'list' – Print out all phone numbers and its message.
        v. 'quit' – Quit the program
    D. Output: The result of each command.
    E. Files to submit:
        i. main.cpp – main() must be in this file.
        ii. message.h – Just copy the following code skeleton.
        iii. message.cpp – Implements MessageBook member functions.
        iv. A CMakeLists.txt to generate the executable

```
$ ./message_book
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
$
```

Code skeleton:

```
class MessageBook {
 public:
  MessageBook();
  ~MessageBook();
  void AddMessage(int number, const std::string& message);
  void DeleteMessage(int number);
  std::vector<int> GetNumbers();
  const std::string& GetMessage(int number);

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

2. Write a program for integer set operations.
    A. Implement functions in the code skeleton.
    B. This program should take user input repeatedly
    C. Input:
        i. { num1 num2 … numk1 } OP { num1 num2 … numk2 }
        ii. OP:
            1. + : Union
            2. * : Intersection
            3. - : Difference
        iii. 0 – Quit the program.
    D. Output: The resultant set of operations.
    E. Files to submit:
        i. main.cpp – main() must be in this file.
        ii. setfunc.h – Function declarations (Just copy the following code skeleton).
        iii. setfunc.cpp – Function definitions.
        iv. A CMakeLists.txt to generate the executable

```
$ ./simple_int_set
{ 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
$
```

Code skeleton:

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