Assignment 4 - STL: Sets, Maps, and Algorithms

- The problems of this assignment must be solved in C++.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320143/2018_1r2/Grading-Criteria-C++.pdf

Problem 4.1 *Reversing a vector* (1 point)

Presence assignment, due by 18:30 h today

Write a program which fills a vector with the integer values from 1 to 30. Then add the value 5 at the end of the vector. Reverse the vector using the `reverse()` function from the `algorithm` library and print the content of the vector on the standard output using an iterator. Then replace all occurrences of the value 5 by the value 129 using the `replace()` function from the `algorithm` library and print the modified vector again on the standard output.

Problem 4.2 *Lotto* (1 point)

Presence assignment, due by 18:30 h today

In the lottery 6 out of 49 numbers are randomly drawn. Draw six different numbers using the formula `rand() % 49 + 1`. Then add the drawn number to a container that stores all drawn numbers (but make sure that your container will not contain duplicates).
After you have drawn all numbers, print them on the standard output in ascending order. Use a suitable STL container that supports the needed operations and corresponding iterators for all actions. Make sure that you initialize the random number generator with the local time of your system at the beginning of your program.

Problem 4.3 *Using sets and multisets* (1 point)

Write a program which does the following using sets and multisets:

1. Creates a set `A` and a multiset `B` able to store integer values.
2. Reads integers from the keyboard until the entered integer is negative. The same non-negative values may repeat in the input.
3. Inserts the non-negative elements into set `A`.
4. Inserts the non-negative elements also into multiset `B`.
5. Prints set `A` and multiset `B` on the standard output (separated by spaces) with an empty line between the elements of `A` and `B`.
6. Prints an empty line on the standard output.
7. Removes all elements with the value 5 from both `A` and `B`.
8. Prints set `A` and multiset `B` on the standard output (separated by spaces) with an empty line between the elements of `A` and `B` as well as after the elements of `B`.
9. Adds the values 14 and 198 to the set `A`.
10. Determines and prints on the screen the union of `A` and `B` (separated by spaces). The result should be a multiset.
11. Determines and prints on the screen the intersection of `A` and `B` (separated by spaces). The result should be a set.

12. Determines and prints on the screen the difference of `A` and `B` (in this order, separated by spaces). The result should be a set.

13. Determines and prints on the screen the symmetric difference of `A` and `B` (in this order, separated by spaces). The result should be a set.

You can assume that the input will be valid.

Problem 4.4 *Another set* (1 point)

A building security system has door locks that are opened by typing a four-digit access code into a keypad. The access code is validated by querying an `Access` object with the following interface:

```
class Access {
  public:
    void activate(unsigned int code);
    void deactivate(unsigned int code);
    bool isactive(unsigned int code) const;
  private:
    // add properties and/or method(s) if necessary
};
```

Each employee is given a different access code, which is activated using the `activate()` method. When an employee leaves the company, his or her access code is deactivated using the `deactivate()` method.
Implement the `Access` class specified above. Write a test program that does the following:

1. Create an `Access` object.

2. Activate the codes 1234, 9999, and 9876.

3. Ask the user for an access code. Read the code from the keyboard.

4. Tell the user whether the door opened successfully.

5. Repeat the last two steps until the door opens.

6. Deactivate the code that worked. Also, deactivate code 9999 and activate code 1111.

7. Repeat steps 3 and 4 until the door opens.

Separate your code into three files: `Access.h` (class definition), `Access.cpp` (implementation of methods) and `testAccessSet.cpp` (test program).
You can assume that the input will be valid.

Problem 4.5 *Using maps* (1 point)

Write a program which creates a database of names and birthday dates. Your program should read the data from a file called "`data.txt`" which contains a name followed by a corresponding birthday date on different lines repeated for many persons. A name consists of first name and last name separated by space. Your program should read the content of the file and use a `map` to store names and birthday dates.
Then "simulate" querying your database (i.e., the map) by asking for a name from the keyboard and printing on the standard output the corresponding birthday date. If the name is not in your container then print "`Name not found!`" on the screen.
You can assume that the input will be valid and the content of the file will be valid if existing.

Problem 4.6 *Another map* (1 point)

The customer that uses the security system described in the previous exercise now wants to associate an access level with each access code. Users with high access levels can open doors to more security-sensitive parts of the building than users with lower access levels. Begin with your solution to **Problem 4.4**. Modify the `Access` class from the previous exercise so that an integer access level is associated with each code. The new interface should be as follows:

```
class Access {
  public:
    void activate(unsigned int code, unsigned int level);
    void deactivate(unsigned int code);
    bool isactive(unsigned int code, unsigned int level) const;
  private:
    // add properties and/or method(s) if necessary
};
```

The `isactive()` method returns `true` if the specified access code has an access level greater than or equal to the specified access level. Of course, it should return `false` if the access code is not active at all. Modify the main program to do the following:

1. Create an `Access` object.
2. Activate code 1234 with access level 1, code 9999 with access level 5, and code 9876 with access level 9.
3. Ask the user for an access code. Read the code from the keyboard.
4. Assuming a door that requires access level 5 for entry, tell the user whether the door opened successfully.
5. Repeat the last two steps until the door opens.
6. Deactivate the code that worked. Change the access level of code 9999 to 8. Activate code 1111 with access level 7.
7. Ask the user for an access code. Read the code from the keyboard.
8. Assuming a door that requires access level 7 for entry, tell the user whether the door opened successfully.
9. Repeat the last two steps until the door opens.

Separate your code into three files: `Access.h` (class definition), `Access.cpp` (implementation of methods) and `testAccessMap.cpp` (test program).
You can assume that the input will be valid.

How to submit your solutions

- Your source code should be properly indented and compile with `g++` without any warnings (You can use `g++ -Wall -o program program.cpp`). Insert suitable comments (not on every line ...) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader.
  Each program **must** include a comment on the top like the following:
  ```
  /*
     CH08-320143
     a4_p1.cpp
     Firstname Lastname
     myemail@jacobs-university.de
  */
  ```
- You have to submit your solutions via *Grader* at
  `https://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `k.lipskoch@jacobs-university.de` **with a subject line that begins with CH08-320143**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Thursday, March 15th, 10:00 h.