

## Assignment 3 - STL: Vectors, Lists, and Deques

- 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/320143/2019\\_1r3/Grading-Criteria-C++.pdf](https://grader.eecs.jacobs-university.de/courses/320143/2019_1r3/Grading-Criteria-C++.pdf)

### Problem 3.1 *Using vectors I*

(1 point)

**Presence assignment, due by 18:30 h today**

Write a program which does the following using a `vector` object:

1. Read words from the keyboard (one per line) until the entered word is equal to the word "stop". You can assume that the words do not contain whitespace.
2. Insert these words into the vector at the end (excluding the word "stop") in the order of their entering.
3. Print the words on the standard output separated by comma using the index operator. Make sure that you do not print a comma after the last word.
4. Print the words on the standard output separated by space using a corresponding iterator.

*You can assume that the input will be valid.*

### Problem 3.2 *Using vectors II*

(1 point)

**Presence assignment, due by 18:30 h today**

Write a program which does the following using a `vector` object:

1. Read strings from the keyboard (one per line) until the word is equal to the string "STOP". Make sure that you can read strings which contain spaces and/or tabs as well.
2. Insert the entered strings into the vector at the end (excluding the string "STOP") in the order of their entering.
3. If the second and fourth elements (i.e., strings) exist, swap the second and fourth element. If one of them or both do not exist then print a message using the standard error stream.
4. Replace the last element with the string "?". If there are no elements in the vector then print a message on the standard error stream.
5. Print the strings on the standard output separated by semicolon using the index operator. Make sure that you do not print a semicolon after the last string.
6. Print the strings on the standard output separated by – using an iterator. Make sure that you do not print a – after the last string.
7. Print the strings on the standard output in the reversed order separated by space using an iterator without reordering the elements of the container.

*You can assume that the input will be valid.*

### Problem 3.3 *Using lists*

(1 point)

Write a program which does the following using two `list` objects:

1. Create two lists (A and B).
2. Read integers from the keyboard until the entered integer is negative or zero.
3. Insert the positive integers into list A by adding to the beginning.

4. Insert the same positive integers into list B by adding to the end.
5. Print list A (separated by spaces) on the standard output and print list B (separated by spaces) in reversed order of the elements into a file called "listB.txt".
6. Print an empty line on the standard output.
7. Move the last element of the lists to the beginning (for both lists).
8. Print list A, print list B on the standard output (both separated by comma) using an iterator. Make sure that you do not print a comma after the last element.
9. Print an empty line on the standard output.
10. Merge list A into list B (considering conditions for merging).
11. Print the result of the merging as a sorted list on the standard output (separated by spaces).

*You can assume that the input will be valid and the input will not result in empty lists.*

### **Problem 3.4** *Using deque*

(1 point)

Write a program which does the following using a deque object:

1. Create a deque A able to store double values.
2. Read double values from the keyboard until the entered double value is 0.
3. Insert the positive elements at the beginning of A and the negative elements at the end of A.
4. Print the elements of A on the standard output separated by spaces.
5. Print an empty line on the standard output.
6. Add 0 into the middle of the deque (between the last positive and before the first negative element).
7. Print the elements of A on the standard output separated by semicolons. Make sure that you do not print a semicolon after the last element.

*You can assume that the input will be valid.*

### **Problem 3.5** *Another deque*

(1 point)

A company that provides a special service for wind surfers installs wind gauges at popular wind-surfing locations. Each gauge reports the current wind speed to a central computer every 6 minutes. When a user connects to the service she or he is able to retrieve recent high, low, and average wind speeds for her or his selected location. The central computer creates a WindGauge object for each gauge. The WindGauge class interface looks as follows:

```
class WindGauge {
    public:
        WindGauge(int period = 10);
        void currentWindSpeed(int speed);
        int highest() const;
        int lowest() const;
        int average() const;
    private:
        // add properties and/or method(s) here
};
```

The constructor argument specifies how much history is retained by the object (default 10 periods which is 1 hour). When `currentWindSpeed()` is called, the current wind speed is added to the history. If the history is then longer than the specified period, the oldest wind speed is discarded. The other three functions return the highest, lowest, and average wind speeds reported during the history period.

Implement the WindGauge class as specified above. Add properties to the class and write a dump function that prints out the lowest, highest, and average wind speed for a WindGauge's data. Write a test program that does the following:

1. Create a `WindGauge` object.
2. Add five wind speeds: 14, 16, 13, 15, and 15, and then dump the gauge.
3. Add eight more measurements: 15, 17, 17, 16, 20, 15, 16, and 20 (bringing the total to over 10) and dump the numbers again.

Separate your code into three files: `WindGauge.h` (class definition), `WindGauge.cpp` (implementation of methods) and `testWindGauge.cpp` (test program).

### Bonus Problem 3.6 *Priority queues*

(1 point)

Use documentation to read about the STL container called `priority_queue`. After consulting documentation write a program which illustrates basic operations on a priority queue. Make sure that you define your own ordering criterion (i.e., do not use the default one).

### 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
  a3_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 coursenummer, 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 Tuesday, April 9<sup>th</sup>, 10:00 AM.**