Lab14a

The aim of the task is to create a tool for student evaluation, which, among others, allows you to check whether the student has passed. The student list is stored in the `list` container, and the student grades are stored in the `vector` container. We assume that there were 10 tasks for 10 points (only integer points). The student has passed if he/she has obtained min. 50% points.

ATTENTION! When solving the task, we use iterators or range-based for loop to operate on containers (in the case of a `vector` container, you cannot use classic indexing).The main function is given and it should not be modified, except for uncommenting subsequent parts of the task.

Part 1 – 1 pts

In the `student` class, implement the necessary constructor, operator<< and the comparison operator. See example usage in main and example output from the program.

Part 2 – 1 pts

In the `evaluation` class, implement the necessary constructor and operator <<. Students are printed to the screen in a lexicographically sorted manner according to the surname and later name. See example usage in main and example output from the program.

Part 3 – 1 pts

In the `student` class, implement the `is_passing` and `show_is_passing` methods. The `is_passing` method checks whether the student has passed. In the implementation, the `for_each` algorithm should be used together with the functionality of counting student points, implemented as a lambda expression. The `show_is_passing` method displays appropriately formatted information about the student, along with information about whether or not he/she passed. See example usage in main and example output from the program.

Part 4 – 1 pts

In the `evaluation` class implement the `show_is_passing` method. The `show_is_passing` method displays appropriately formatted information about all students . See example usage in main and example output from the program. The resulting list is arranged in the original order of addition.

Part 5 – 1 pts

In the `evaluation` class implement the `clear_not_passing` method. The `clear_not_passing` method erases from the student list all students which didn't pass. In the implementation, the `erase` method from `list` container should be used. See example usage in main and example output from the program.

Part 6 – 0,5 pts

In the `student` class, implement the method `is_leader`, which checks whether the student is a leader. A student is a leader if he/she has obtained at least as many maximum marks as indicated by the `level`. In the implementation, the `count_if` algorithm should be used together with properly defined lambda expression. See example usage in main and example output from the program.

Part 7 – 1 pts

In the `evaluation` class implement the `save_to_file` method. The `save_to_file` method saves information about the students to the file. Additionally, at the beginning of the file the information about number of leaders is also printed. In the implementation, the `count_if` algorithm should be used together with properly defined functor `check_leader`. See example usage in main and example output from the file.

Part 8 – 1,5 pts

In the `student` and `evaluation` classes implement the `histogram` method. Histogram Is realized in form of map container. The key of each map element represents the possible grades, and the values store the number of occurrences of that grade. Method `histogram` in `evaluation` class defines and initialize histogram, than fills histogram with occurrence of grades from each student invoking method `histogram` from `student` class. Finally, histogram is printed to the screen. See example usage in main and example output from the program.

Example program output:

```
1:
test 1: 7 7 8 9 6 10 10 10 8 9
test 2: 2 2 4 7 4 3 1 2 5 7

test_1 < test2 - TRUE
test_2 < test1 - FALSE

2:
Ariola Evelin: 10 6 7 4 7 8 6 10 4 7
Ariola Troy: 2 2 4 7 4 3 1 2 5 7
Mcguckin Emile: 7 7 8 9 6 10 10 10 8 9
Pulver Ji: 7 3 8 5 6 6 10 6 3 9
Townes Shakira: 3 5 3 6 7 2 3 4 2 1

3:
test 1: passing
test 2: not passing
```

4:
Pulver Ji: passing
Ariola Troy: not passing
Ariola Evelin: passing
Mcguckin Emile: passing
Townes Shakira: not passing

5:
Ariola Evelin: 10 6 7 4 7 8 6 10 4 7
Mcguckin Emile: 7 7 8 9 6 10 10 10 8 9
Pulver Ji: 7 3 8 5 6 6 10 6 3 9

6:
test 1: 7 7 8 9 6 10 10 10 8 9 - LEADER!
test 1: 7 7 8 9 6 10 10 10 8 9 - NOT LEADER!

7:
Check file!

8:
pts 0: N = 0
pts 1: N = 0
pts 2: N = 0
pts 3: N = 2
pts 4: N = 2
pts 5: N = 1
pts 6: N = 6
pts 7: N = 6
pts 8: N = 4
pts 9: N = 3
pts 10: N = 6

Example file output:

Number of students with more than 2 maximum grades: 2
Ariola Evelin: 10 6 7 4 7 8 6 10 4 7
Mcguckin Emile: 7 7 8 9 6 10 10 10 8 9
Pulver Ji: 7 3 8 5 6 6 10 6 3 9