Assignment 7.

Standard template library and data processing

Purpose of the exercise

This exercise will help you do the following:

- Practice use of STL templates and algorithms.
- Practice working with STL containers.
- Practice use of callable objects and predicates in processing data within containers.

Requirements

Your task is to implement the *solution.h* file that defines 5 missing functions used by the driver:

- `print_file_names(const file_records& map);`
- `print_non_empty_files(const file_records& map);`
- `print_empty_files(const file_records& map);`
- `get_parameters(file_records& map);`
- `remove_empty(file_records& map);`

There are a couple of challenges in solving this puzzle. Firstly, you have to figure out the purpose, parameters and the return data types of each function based on the driver code. Secondly, you must observe the following restrictions. You are not allowed to:

- Include any headers.
- Define any other functions than the ones mentioned above.
- Define any new complex types or templates; type aliases are permitted.
- Use lambda expressions.
- Use the following operators:
  - `.` (member access),
  - `->` (member access via a pointer),
  - `*` (dereference).
- Use explicit iteration (`for`, `while`, `do while`) or selection (`if`, `switch`, `?:`) statements or operators. You can use functions or function templates (such as `std::for_each`) to iterate over a collection of elements.
- Use `std::cout`, `std::cerr` or any other functions that perform printing of text to the console; you have to use a provided function to do it.
- Use a keyword `auto`.

To compile and test your code, simply execute the following command:

```
g++ -Wall -Werror -Wextra -Wconversion -pedantic -std=c++17 -o main main.cpp
main > actual-output.txt
diff actual-output.txt expected-output.txt --strip-trailing-cr
```

Make sure that the resulting file *actual-output.txt* matches the provided file *expected-output.txt*.

Study the templates available in the <functional>, <algorithm>, and <utility> headers before implementing any code for this assignment. You can use STL templates presented in the classroom, but also ones that have not been covered that you discovered yourself.

You may find the following templates useful:

- `std::begin()`, `std::end()`, `std::size()`
- `std::back_insert_iterator`, `std::back_inserter()`
- `std::bind()`
- `std::copy()`, `std::copy_if()`
- `std::count()`, `std::count_if()`
- `std::reference_wrapper`, `std::ref()`, `std::cref()`
- `std::move()`, `std::forward()`, `std::forward_as_tuple()`
- `std::transform()`
- `std::tuple`, `std::make_tuple()`

Requested files

Without any comments you can expect the file to be around the following size:

- *solution.h* - 60 lines.

No *checklist* or comments are required; it is a good practice to include them, but the automated grading tests will not penalize you for the lack of comments, at least in this assignment.

At this level of the course, it is expected that the style of your code is elegant, easy to read and has a desired quality of being self-explanatory and easy to understand. The automated grading tests will not penalize you for low quality of the code working properly, at least in this assignment.

Submitting the deliverables

You have to upload requested files to Moodle - DigiPen (Singapore) online learning management system, where they will be automatically evaluated.

To submit your solution, open your preferred web browser and navigate to the Moodle course page (pay attention to the section name suffix at the end of the course name). In the course page find a link to the Virtual Programming Lab activity that you are submitting.

In the **Description** tab of the activity you can find a due date, and a list of requested files. When you switch to the **Submission** tab you should see the controls for uploading or typing exactly the file that is required. Upon clicking the *Submit* button the page will validate a submitted file and report any errors. If the submission was successful, the page will display a message that the submission has been *Saved*. You should press the *Continue* button to see the *Submission view* page with the results of the evaluation and the grade.

If you received an *A* grade, congratulations! If not, before the due date you can still review and update your solution and resubmit again. Apart from exceptional circumstances, all grades after the due date are final, and students who did not submit their work will receive a grade *F*.