Assignment 1.

Refresher on object-oriented programming

Purpose of the exercise

This exercise will help you do the following

1. Develop familiarity with the Virtual Programming Lab activity and automated grading of submissions in *Moodle* used throughout the course.
2. Refresh your knowledge about object-oriented programming, in particular encapsulation, and dynamic memory allocation.
3. Improve your fluency in reading, interpreting and writing constructors, destructors, member functions, and data member declarations.

Requirements

Suppose that a class `Background` is a container for storing colors; a one-dimensional container to be exact. Each `Color` object represents a value of an RGB pixel (with red, green and blue components).

Your task is to implement a class `Background` (with the declaration of member functions in *Background.h*, and their definitions in *Background.cpp*), and complete the implementation of a class `Color` (a header file *Color.h* has been provided for you, but you must write all the code necessary to define all its member functions in *Color.h*).

While implementing require code take into consideration using proper syntax (i.e. constructor's initialize lists), `const`-correctness of member functions and parameters, proper encapsulation, and other aspects of object-oriented programming covered in *[CS 170]*.

The important part of this exercise is taking care of proper dynamic memory management, as a `Background` object must meet the requirement of being exactly 16 bytes in size (take note that the code will be compiled in 64-bit environment). Ensure that there are no leaks, no double-delete, no dangling pointer related problems, and copies are created using deep-copy mechanism to ensure that a `Background` object is the sole owner of `Color` objects it stores.

You are not allowed to include any additional headers, besides the ones provided or required in this exercise, with an exception of the following standard headers (they are permitted, but not required):

- `<cstddef>`
- `<algorithm>`

To properly complete the assignment, get familiar with the test cases defined in *main.cpp*:

- Test cases 1-6 should be easy to understand by just reading the code.
- Test cases 7-10 use expression that have not been covered, but based on their brief description you should quickly figure out what it their purpose.

Lastly, test the code in g++ with the following flags before submission:

```
g++ -Wall -Werror -Wextra -Wconversion -pedantic -std=c++17 -o main.exe
main.cpp Background.cpp Color.cpp
```

Requested files

As this is a warm-up exercise only minimal code is requested. Without any comments you can expect files to be around the following sizes:

- *Background.cpp* - 50 lines,
- *Background.h* - 35 lines,
- *Color.cpp* - 30 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

Every assignment specification, like this one, will instruct you what deliverables are expected as a result of the assignment. You have to upload these 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 course on Moodle (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 files that are required. Upon clicking the *Submit* button the page will validate submitted files and report any errors. If the submission was successful, the page will display a message that the submission has been *Saved*. You can 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*.