CSCI204/MCS9204/CSCI804
Object and Generic Programming in C++
Laboratory Exercise 8 (Week 10)

Task One: I/O Manipulators (0.3 marks)

Define a manipulator *format* in a file **task1.cpp**. The manipulator *format* can take three arguments: string of base (such as “oct”, “dec” and “hex”), the length that an integer can be displayed, and a character can be used to pad the output value when the length of the value is not long enough. Write a driver program include `main()` function in the file **task1.cpp** to test the manipular. For example:

```
cout << format("oct", 10, '#') << 20 << endl;
```

The result is

```
########24
```

Change the format like

```
cout << format("hex", 8, '!') << 20 << endl;
```

The result is

```
!!!!!!14
```

Change the format like

```
cout << format("dec", 12, '@') << 20 << endl;
```

The result is

```
@@@@@@@@@@20
```

Task Two: A PGM Reader (0.7 Marks)

In this task, you are required to develop a program, **pgmReader**, to read a gray-scale image saved in `pgm` format (**PGM**, Portable Gray Map). The task should be completed in two steps:

(a) Design and implementation of class `Image` as described below.
(b) Design and implement the program, **pgmReader,** as specified below**.**

An image consists of a grid of pixels (picture elements). For a gray-scale image, each pixel has one value representing its luminance or brightness. The pixel value usually ranges from 0 to 255, with 0 being the darkest (black) and 255 being the brightest (white).

PGM is a simple file format to store gray-scale images. You are required to design and develop a class, **Image**, that is able to read a gray scale image from a PGM image file and report the following statistics of the image: `width, height, minimum intensity, maximum intensity, and average intensity`. You are required to declare the **Image** class in **Image.h** and place its implementation in **Image.cpp.** Read the appendix for the specifications of PGM.

Using the **Image** class, design and implemented a main program, **pgmReader.cpp,** that reads an image stored in a PGM format file, calculates and outputs the statistics. Following are examples that show how your program may be used and marked. Note: **$** is assumed to be the prompt of a UNIX terminal.

```
$ pgmReader c01.ppm
<c01.ppm> is not a PGM file
```

```
$ pgmReader c01.pgm
Read a gray-scale image from a PGM file <c01.pgm>.
Statistics of the image:
Width:         1024
Height:        768
Min-Intensity: 0
Max-Intensity: 255
Ave-Intensity: 131.95
```

The compile directive would be something like

```
g++ –o pgmReader pgmReader.cpp Image.cpp
```

Two PGM images are provided for testing your program.

Submission:

You should submit the files of tasks to the server by **11:59 PM on Friday, 13 May 2016** via command:

```
submit –u your-user-name –c CSCI204 –a L8  task1.cpp Image.h Image.cpp pgmReader.cpp
```

and input your password.

Make sure that you use the correct file names. The UNIX system is case sensitive. You must submit all files in one *submit* command line.

After submit your assignment successfully, please check your email of confirmation. You should keep this email for the reference.

You would receive ZERO of the marks if your program codes could not be compiled correctly.

Later submission will not be accepted. Submission via e-mail is NOT acceptable.

End of Specification

Appendix: PGM Format Specification

pgm

The PGM format is designed to be extremely easy to learn and write programs for. A PGM image represents a grayscale graphic image. For most purposes, a PGM image can just be thought of an array of arbitrary integers, and all the programs in the world that think they're processing a grayscale image can easily be tricked into processing something else.

The name "PGM" is an acronym derived from "Portable Gray Map." Each PGM image consists of the following:

1. A "magic number" for identifying the file type. A pgm image's magic number is the two characters "P5".
2. Whitespace (blanks, TABs, CRs, LFs).
3. A width, formatted as ASCII characters in decimal.
4. Whitespace.
5. A height, again in ASCII decimal.
6. Whitespace.
7. The maximum gray value (Maxval), again in ASCII decimal. Must be less than 65536, and more than zero.
8. Newline or other single whitespace character.
9. A raster of `Height` rows, in order from top to bottom. Each row consists of `Width` gray values, in order from left to right. Each gray value is a number from 0 through Maxval, with 0 being black and Maxval being white. Each gray value is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.

   A row of an image is horizontal. A column is vertical. The pixels in the image are square and contiguous.

10. Each gray value is a number proportional to the intensity of the pixel. A value of zero is black. A value of Maxval represents white and the most intense value in the image and any other image to which the image might be compared.
11. Characters from a "#" to the next end-of-line, before the maxval line, are comments and are ignore.