Programming Design, Spring 2016
Homework 3

Instructor: Ling-Chieh Kung
Department of Information Management
National Taiwan University

Please upload one PDF file for Problem 1 and two CPP files for Problems 2 and 3 to PDOGS at http://pdogs.ntu.im/judge/. Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***2:00 am, March 14, 2016***. Please answer in either English or Chinese.

Before you start, please read Section 3.10–3.12 and 4.1–4.11 of the textbook.[1]

The TA who generated the testing data and will grade this homework is ***Chien Huang***.

Problem 1

(20 points; 5 points each) In this problem, we will discuss the multiplication table.

(a) Consider the following program that generates an $n \times n$ multiplication table:

```
#include <iostream>
using namespace std;

int main()
{
  int n = 0;
  cin >> n;

  for(int i = 1; i <= n; i++)
  {
    for(int j = 1; j <= n; j++)
      cout << " " << i * j;
    cout << "\n";
  }
  return 0;
}
```

Rewrite it by using `while` only. Comment on which implementation is better and why.

(b) Continue from Part (a). Alice suggests to modify the inner block to the following:

```
    for(int j = 1; j <= n; j++)
    {
      if(i * j < 10)
        cout << "  ";
      else if(i * j < 100)
        cout << " ";

      cout << " " << i * j;
    }
```

Explain the impact of the added `if-else` statement and how it works.

(c) Continue from Part (a). Bob suggests to modify the program to the following:

[1] The textbook is *C++ How to Program: Late Objects Version* by Deitel and Deitel, seventh edition.

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
  int n = 0;
  cin >> n;

  cout << setw(4) << " " << " | ";

  for(int i = 1; i <= n; i++)
    cout << setw(4) << i;

  cout << "\n-----+-------------------\n";

  for(int i = 1; i <= n; i++)
  {
    cout << setw(4) << i << " | ";
    for(int j = 1; j <= n; j++)
      cout << setw(4) << i * j;
    cout << "\n";
  }
  return 0;
}
```

Search “C++ setw” or similar keywords online to teach yourself what the function `setw(4)` does. Briefly explain it.

(d) Continue from Part (c). Note that the statement

```cpp
  cout << "\n-----+-------------------\n";
```

is not good because the line may be too long or too short when `n` is small or large. Modify this statement (and probably expand it to multiple statements) to generate a line whose length is appropriate for any value of `n`.

Problem 2

(40 points) A factory makes color printings according to customers' requests. When a customer makes an order, she/he specifies the color she/he wants in RGB color code.[2] To do so, she/he specifies a three-dimensional vector (r, g, b), where r, g, and b are three integers in 0 and 255, one for red, one for green, and one for blue. She/he should also specify the number of pieces of papers to be printed. The factory then prints out that number of copies by trying to follow the RGB requirement.

Unfortunately, the printing machine is not perfect and the printed color may deviate from the color specified by the customer. Therefore, the factory checks the printed colors before shipping the products to a customer. The contract says that the printed color may deviate from the specified color by at most k in each dimension. If too many copies violate the requirement, the factory needs to reprint. For example, suppose a customer specifies a color as $(102, 102, 255)$ (which is a kind of light blue) when $k = 2$, then a printed color $(100, 104, 253)$ is acceptable while $(102, 102, 252)$ is not.

Now, you are given a specified color, n copies with different printed colors, and the quality standard k. Your task is to find out the number of unacceptable copies.

[2] For more details about the RGB color model, please see `https://en.wikipedia.org/wiki/RGB_color_model`.

Input/output formats

There are 15 input files. In each file, there are $n + 2$ lines of input. The first line contains three integers r, g, and b, all within 0 and 255. The second line contains two integers $n \in \{1, 2, ..., 1000\}$ and $k \in \{0, 1, 2, 3, 4, 5\}$. The ith line, $i = 3, 4, ..., n + 2$, contains three integers r_i, g_i, and b_i, all within 0 and 255. Two consecutive integers in each line are separated by one white space. Given these input values, your program should print out one integer as the number of copies that violate the quality standard.

What should be in your source file

Your .cpp source file should contain C++ codes that will both read testing data and complete the above task. For this problem, you are allowed to use techniques covered in lectures and Chapters 1 to 4 in the textbook. NO other techniques are allowed. Finally, you should write relevant comments for your codes.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 10 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.

Problem 3

(40 points) Continue from the previous problem. Because you did a good job in identifying bad copies, the factory owner now asks you to help improve the printing process. If particular, she/he wants to find out the *error tendency* and *error correlation*:

- Error tendency measures whether the errors in each color are random, i.e., sometimes too large and sometimes too small. For example, if the customer specifies r and for a printed color $r_i < r$, we say the error is *negative*; otherwise, it is *positive*. For each color, if the number of positive and negative errors differ too much, there may be a systematic error in the printing machine. Therefore, if we define
$$\text{error tendency} = \text{number of positive error} - \text{number of negative error},$$
we should prefer the error tendency to be close to 0.
- Error correlation measures whether the printing of three primitive colors (R, G, and B) tends to be erroneous at the same time. For each printed color, the number of primitive colors that are erroneous is 0, 1, 2, or 3. If for most of the bad copies the number of erroneous primitive colors are 2 or 3, we can say that these errors tend to occur together. Therefore, we define
$$\text{error correlation} = \frac{\text{number of bad copies with 2 or 3 erroneous primitive colors}}{\text{number of bad copies}}$$
and prefer the error correlation to be close to 0, if there is any bad copy.

For both measurements, we count only bad copies, i.e., those copies with at least one primitive color that deviates from the specified primitive color by more than k.

As an example, suppose the specified color is $(120, 120, 120)$, $k = 1$, and three printed colors are $(122, 120, 120)$, $(120, 121, 120)$, and $(122, 109, 121)$, then we have:

- The error tendencies of red, green, and blue are 2, -1, and 1, respectively. Note that the error tendency of green is -1 rather than 0 because $(120, 121, 120)$ is a good copy and should not be counted.

- The error correlation is $\frac{1}{2}$. Note that the denominator is 2, the number of bad copies, instead of 3, the number of all copies.

Now, given you are given a specified color, n copies with different printed colors, and the quality standard k. Your task is to calculate the error tendencies of the three primitive colors and the error correlation.

Input/output formats

There are 15 input files. The input format is exactly the same as that in the previous problem. Given these input values, your program should print out five integers in order: the error tendency of red, error tendency of green, error tendency of blue, number of bad copies with 2 or 3 erroneous primitive colors, and number of all bad copies.

Grading criteria

- 30 points will be based on the correctness of your output. PDOGS will compile your program, feed testing data into your program, and check the correctness of your outputs. Each fully correct set of outputs gives you 2 points.
- 10 points will be based on how you write your program, including the logic and format. Please try to write a robust, efficient, and easy-to-read program.