Programming Design, Spring 2014
Homework 1

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

Submission. To submit your work, please upload the following two files to the online grading system at http://lckung.im.ntu.edu.tw/PD/.

1. A .pdf for Problems 1 to 3.
2. Your .cpp file(s) for Problems 4 and 5 (optional).

Each student must submit her/his individual work. No hard copy. No late submission. The due time of this homework is ***8:00am, February 24, 2014***. Please answer in either English or Chinese.

Problem 0

(0 points) Please read the last section "Formatting a C++ program" of the slides used on February 17 by yourself. Please also read Sections 1.7–1.9, 1.14, 2.1–2.8, 3.1–3.5, and 3.7–3.9 of the textbook.[1]

Problem 1

(5 points) Use your own words to explain what are the differences among machine languages, assembly languages, and high-level languages.

Problem 2

(5 points) Use your own words to explain what are the differences between the assignment operator `=` and comparison operator `==`.

Problem 3

(15 points) Answer this program according to the following program. You may assume that the user will always enter an integer no greater than 10^5.[2]

```
int candidate = 0, divisor = 2;
bool isPrime = true;
cin >> candidate;

while(divisor < candidate)
{
    if(candidate % divisor == 0)
    {
        isPrime = false;
        break;
    }
    divisor = divisor + 1;
}
cout << isPrime;
```

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

[2]Though not required, the `if-else` statement is useful for Parts b and c. This technique will be officially introduced in class, but you are encouraged to study it by yourself and use it now.

(a) (5 points) Use your own words to explain what this program does.

(b) (5 points) Explain what will happen if a negative integer is entered. Then insert some codes into this program to output "`Only integers greater than 2 are accepted.`" and then skip the examination done in the given program.

(c) (5 points) Replace the `cout` statement to display "`The input number is a prime number.`" when it is a prime number or "`The input number is a common number.`" otherwise.

Problem 4

(75 points) In this problem, we write a C++ program as a divisor enumerator.[3] Your program should allow the user to repeatedly input a positive integer and then list all the divisors of the integer. For example, if the input is 126, the answer should include 1, 2, 3, 6, 7, 9, 14, 18, 21, 42, 63, and 126. You may assume that the user will only enter integers no greater than 10^5.

One key feature of your program is that it should accept repeated inputs. The detailed rule is:

- If a positive integer is entered by the user, the program should output all the divisors separated by white spaces,[4] change to a newline, and then ask the user to enter another number. In particular, if the input number is a prime number, the program should output 1 and that number and then change to a newline.
- If zero is entered, the program should terminate immediately.
- In other cases, the program should output a warning message, change to a new line, and then wait for the user's next input.

For all messages used as prompts, please design them by yourself so that clear instructions are given to the user.

What should be in your source file

Your .cpp source file should contain C++ codes that will complete the above task. For this problem, you are NOT allowed to use techniques not covered in lectures. Finally, you should write relevant comments for your codes.[5] This will be particularly useful when we come back to this program again, which will happen very soon.

Grading criteria

- 70% of your grades for this program will be based on the correctness of your output. The TAs will compile your program, run your program, and input several testing data to test your program. They will then determine the correctness of your program.
- 30% of your grades for this program 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.

Bonus: Problem 5

(20 points) Please write another C++ program (with any technique you have) that works similarly to that in Problem 4. However, now for each input, list all its *prime* divisors. For example, if the input is 126, the output should be 2, 3, and 7. The grading criteria is the same as Problem 4.

[3]Though you will upload your program to the online grading system, the system will start to grade your program starting from Homework 2. This is to help you eliminate the tedious works regarding the formats of input and output so that you may concentrate on writing your first program.

[4]For example, to output 2, 9, 11, and 13, your program should output "`2 9 11 13`".

[5]See the last section "Formatting a C++ program" for an introduction to comments.