Programming Design, Spring 2013
Homework 03

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

To submit your work, please upload the following two files to the online grading system PDOGS at http://stella.im.ntu.edu.tw/online-judgement/.

1. A PDF file (a .pdf file) for Problems 1 to 2.
2. Your source file (the .cpp file) for Problem 3.

NO hard copy and NO late submission. The due time of this homework is ***1:00pm, March 11, 2013***.

Problem 1

(5 points) Consider the following program

```
int a = 0;
cin >> a;

if(a = 1)
    cout << "a is one!\n";
else
    cout << "a is not one!\n";
```

When will `a is one!` be printed out, if possible? When will `a is not one!` be printed out, if possible?

Problem 2

(25 points) Consider the following program implemented with a `while` statement:

```
int i = 0;
cin >> i;

while(i > 1)
{
    if(i % 2 == 0)
        i = i / 2;
    else
        i = 3 * i + 1;
    cout << i << " ";
}
cout << "the final result: " << i;
```

(a) (10 points) Explain what this program does.

(b) (5 points) If the last line `cout << "the final result:  " << i;` is executed, what may be printed out? Does that depends on the input value of `i`? If so, how does it depend on the input of `i`? If not, why?

(c) (10 points) Rewrite the program with a `for` statement.

Note. The question "Is there a positive integer that does not go to 1 through the above process?" is typically called the Collatz Problem. You may find a lot of discussions on it online. What would be your answer?

Problem 3

(70 points) Please write a C++ program according to the following instructions.

Note. On the online grading system PDOGS, this program is called "PD001". Please submit this program in the PD001 section.

What should your program do

Your program should allow two users to play the game "tic-tac-toe". In a tic-tac-toe game, there are two players, 1 and 2, player on a three by three chessboard (i.e., there are nine cells in total). Each of them takes turns to select an unoccupied cell as her territory. Once one player's three territories make a horizontal, vertical, or diagonal straight line, she wins the game. If no player can do so after all the nine cells are occupied, the game ends with a tie.

Each of the two players selects a cell by indicating an integer number from 1 to 9, where 1 means the left-top cell, 2 means the center-top cell, 3 means the right-top cell, 4 means the left-middle cell, ..., and 9 means the right-bottom cell. This way of labeling the cells immediately shows that to win the game one needs to occupy one of the following eight sets of cells:

- Horizontal straight lines: $\{1, 2, 3\}$, $\{4, 5, 6\}$, $\{7, 8, 9\}$.
- Vertical straight lines: $\{1, 4, 7\}$, $\{2, 5, 8\}$, $\{3, 6, 9\}$.
- Diagonal straight lines: $\{1, 5, 9\}$, $\{3, 5, 7\}$.

Your program will take nine integers from 1 to 9 as input. These nine integers will be put in one line, separated by eight white spaces. Each number, which appears exactly once, means the cell selected by a player. In this game, player 1 goes first. Let's consider the following example input:

```
1 6 7 4 5 9 2 3 8
```

This means player 1 first selects cell 1, then player 2 selects cell 6, then player 1 selects cell 7, ..., and finally player 1 selects cell 8. Note that even though in an input line there are always nine numbers, the game may end in fewer then nine selections. For example, in the above example, the game ends when player 2 selects cell 3 in the eighth step.

Given a line of input, your program should output two things. First, a number indicating who wins, where 0 means a tie, 1 means player 1, and 2 means player 2. Second, an integer between 1 and 9 indicating after how many selections does this game end. The two numbers should be separated by a white space. A new line character (i.e., `\n`) should then be appended after the second number. For example, in the above example, the output should be

```
2 8
```

with a new line character appended. The files "PDSp13_hw03_testing.txt" and "PDSp13_hw03_result.txt" provides ten example games.

What should be in your source file

Your .cpp source file should contain C++ codes that will complete the above task. Moreover, you should write relevant comments for your codes.

While you are developing your program, you certainly wants to output the progress of the game to help you understand your program. At the beginning stage, you may also use different ways to input the selection of cells. Nevertheless, once you feel that the logic of your program is done and you want to upload your program for an online grading, please carefully follow the rules of input and output specified above. In particular, all the codes printing out the game progress should be commented. As the TAs

will still open your file to check the readability of your program, please DELETE (not just commented) all irrelevant codes in your final submission.

Grading criteria

- 70% of your grades for this program will be based on the correctness of your output. The online grading system will input 35 sets of testing data and then check your outputs. You may only see the grades of running your program on these data but cannot see the inputs and outputs. These 35 sets count for 70 points, i.e., 2 points for each set.
- 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.

Things you need to do for online grading

The online grading system input 35 sets of data into your program one by one and examine your outputs accordingly. As each set of data contains nine integers, In the testing data, there are 315 integers, separated by 314 white spaces or new line characters. For your program to process all these data automatically, your program must be written in a way that it allows multiple games. More precisely, your program should contain a `while` loop whose looping condition is an `cin` statement. It would be something like

```
int currentChoice = 0;

while(cin >> currentChoice)
{
    ...
}
```

This loop will ask the user (i.e., the grading system) to input one integer as the current player's current choice. It will keep looping until the testing data is finished. The codes that determine who wins the game should all be put inside the above `while` loop.

Do not forget that nine integers together form one game! If your program determines that one game ends before all the nine integers are examined, you need to discard the remaining integers in that game by `cin` them into some variables that are not used at all.

Once your complete your program, you may test your program in the following two ways (before you upload it into the online grading system):

1. You may simply execute the program and type in data with your keyboard.
   (a) If you type in one game in one line (9 integers with 8 white spaces and one "enter"), you should see the result of this game immediately and the program waiting for your next input.
   (b) If you type in two game in one line (18 integers with 17 white spaces and one "enter"), you should see the results of two games in two lines and the program also waiting for your next input.

   If you test your program in this way, naturally your program never ends by itself.
2. You may input the public testing data file (provided with the homework as a .txt file) into your program. To do so, in the console mode, type `hw3.exe < test.txt`, where `hw3.exe` should be replaced by the name of the executable file generated by your C++ file and `test.txt` should be replaced by the file name of the testing data. Then you should see the results of all games. The program then terminates.