**Assignment #5**

**Introduction to C Programming – COP 3223**

**Objectives**

1. To learn how to read input from a file
2. To reinforce knowledge of loops and conditionals

**Introduction: Ultimate Computing Fun**

A new game store is opening in your area! Ultimate Computing Fun (UCF) Games is looking for a home for their wide variety of computer games and has decided to make you a part of their growing business.

Now that the store is up and running UCF Games is looking in a new direction. They would like your assistance in creating an in-house game to increase revenue.

**Problem: Golf Game Scorer (golf.c)**

Research has indicated that sports simulators are expected to be big profit makers this quarter. UCF Games would like to create a new Golf game using in-hourse resources to increase their profits. Other team members will be working on most of the game play and graphics. UCF Games needs your help working on the scoring system.

“The goal of golf is to strike a ball until it drops into a hole in the ground. The fewer times you need to hit the ball to achieve the goal, the better. A game of golf has 18 holes, and on each hole, a particular score (usually 3, 4 or 5) is designated as “par”. A particular player’s score for a hole is simply the number of times they need to hit the golf ball from the starting position to get it into the hole.” (Arup Guha)

The final score is calculated by subtracting the par value from each user’s score. For example, if the user needed 100 strokes to finish 18 holes, and the total par for the 18 holes is 90, their score is 10. Likewise, if the user only needed 80 and the total par was 90, their score is -10. In golf, lower numbers indicate better scores.

Up to 4 users can play a single game of golf. Their scores for each individual hole will be recorded in a file along with the par for each hole. This file will be the input for your scoring system. Your scoring system should calculate the final score for each of the players and output this value to the screen along with the winner of the quartet.

**Recommendations**

While it is not strictly necessary to use arrays to solve this problem, using an array will help you practice for future assignments and exams.

**Input Specification**

There will be no input from the user. All input will come from the input file “input.txt”. A sample input file is available below.

The first line of the file will contain a single positive integer, n (1 <= n <= 4), specifying the number of players.

The second line of the file will contain 18 positive integers that represent the par for each of the holes, from hole 1 to hole 18.

The following n lines will contain 18 positive integers, representing each user’s individual scores for each hole.

**Output Specification**

Output to the screen.

For each of the n players, output a line with the following format:

Player X scored Y.

where X is the number of the player (1-4) and Y is their final score.

At the end of the program, output which of the players in the winner in the following format:

Player X is the winner!

**Sample Run**

Below is a sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above.

**Sample Input File (input.txt)**

2

3 4 4 4 4 5 5 3 3 4 4 4 4 4 5 3 3 4

4 4 5 4 6 4 5 6 3 4 4 4 5 4 3 3 5 4

2 3 4 4 4 5 5 3 3 4 4 4 4 4 4 3 3 4

**Sample Output**

**Player 1 scored 7.**

**Player 2 scored -3.**

**Player 2 is the winner!**

**Deliverables**

One source file – *golf.c* – is to be submitted over WebCourses.

**Restrictions**

Although you may use other compilers, your program must compile and run using Code::Blocks. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

**Grading Details**

Your programs will be graded upon the following criteria:

1) Your correctness

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility – You must submit C source files that can be compiled and executed in a standard C Development Environment. If your program does not compile, you will get a sizable deduction from your grade.