Finalizing Project 2

Part 1. Requirements. List all your requirements for the project. These should be listed in one consolidated list.

Part 2. TDD Development. Relist your requirements in the order that you coded them, with the test cases you used to develop the software listed with that requirement.

Part 3. Regression Test Plan. Include the test cases that could be used to test the system in the future after later enhancements or changes were made. You should also run this test plan against your final coded system.

Part 4. Closing Thoughts. Feel free to share any conclusions you had summarizing what you learned from this assignment, with particular attention paid to the TDD methodology. Did the TDD methodology help you, or hinder you? Also, if you made some fundamental missteps early on, this is a good place to discuss what you have learned from those mistakes.

Part 5: Your code listing. Yes, I want to see your final code. Remember, though, once I see code, I assume nothing else follows, so make sure you put this last.

Some examples and hints:

Your requirements and test cases should be written and documented at a low level of abstraction. Don't assume the reader has memorized all the scoring rules for this fantasy football league.

Don't just paste the league scoring rules into your documentation and try to pass that off as a "requirement."

It is often helpful to number your requirements and test cases. That said, the test case numbering in Part 2 need not match the test case numbering in Part 3.

You should have used boundary testing for many of the test cases in Part 2.

At some point in your development effort, inputs for your test cases should be in the same format as they are given in a box score. Box score data is the program's input.

Many of the requirements in Part 2 may have partial functionality, but all of the test cases for Part 3 will have full functionality.

Ideally, by the time your project is done, you should be able a read in a full box score with players from two teams and calculate the right amount of points. However, if you never get your code working, you can still earn a healthy chunk of partial credit with a well-documented test plan showing how you would have tested the system had you got it working.

The next couple pages have some examples for Parts 1 through 4.

Here are some examples of requirements I might find in Part 1:

1. When processing passer data, the system shall award 4.5 points to the player for every TD pass thrown.

2. When processing rushing and receiving data, the system shall award 6 points for every TD carry or reception.

3. The system shall award two bonus points to any kicker who successfully kicks a field goal of 50 yards or more.

Here is an example of how one of those requirements might be broken down in Part 2:

Requirement 3: *The system shall award two bonus points to any kicker who successfully kicks a field goal of 50 yards or more*.

Test Case 3.1
Input: Player FG Pct Lng Pts 1 John Smith 2/3 67.7 50 5
Expected Output: 1 John Smith has scored 8 points.
Actual Output: 1 John Smith has scored 6 points. (first run) 1 John Smith has scored 8 points. (second run)
Pass/Fail: *Fail and then Pass*
Tester notes: *The program was initially set to check field goals greater than 50 yards, not greater than or equal to, so the test failed the first time around.*

Note: In this example, I am assuming this iteration of TDD was performed after the logic for successful field goals was implemented, but before the logic for missed field goals and extra points had been implemented, which is why the player's expected output is only 8 points.

Here is an example of how kicker data might be tested in the regression test in Part 3:

Test Case 1 - Testing Kicker Data

Input:

When prompted for a box score text file, enter `RAMS-PATRIOTS-KICKERS.txt`

Note: This file contains the following lines:

```
PATRIOTS
KICKING
Player               FG   Pct       Lng   Pts
1    John Smith      2/3  67.7      50    7

RAMS
KICKING
Player               FG   Pct       Lng   Pts
99   James Jones     3/3  100.0     49    11
```

Expected Output:

```
 1 John Smith (Patriots) has scored 8.6 points.
99 James Jones (Rams) has scored 11.0 points.
```

Scoring Notes:

John Smith: 6+1+2-0.4 = **8.6**
(6 pts for 2 FGs; 1 point for 1 XP; 2-pt bonus for 50 yd FG; -0.4 pts for 1 missed FG)

James Jones: 9 + 2 = **11**
(9 pts for 3FG; 2 pts for 2 XPs)

Actual Output:

```
1 John Smith (Patriots) has scored 8.6 points.
99 James Jones (Rams) has scored 11.0 points.
```

Pass/Fail: *Pass*

Tester notes:

Notes:

Notice how there is no need to go open the test file to understand why the expected output contains the values that it has. This is a thoroughly documented test case. Compare it to the one on the next page, which is more sloppily documented.

Depending on how your box scores are set up, it's entirely possible that you can do all your regression testing with just one or two box scores, and therefore one or two test cases (although the expected output for each might be quite lengthy).

Test Kickers
Input: When prompted for a box score text file, enter `KICKERS.txt`

Expected Output:
 	
`1 John Smith (Patriots) has scored 8.6 points.`
`99 James Jones (Rams) has scored 11.0 points.`

Actual Output:
 	
`1 John Smith (Patriots) has scored 8.6 points.`
`99 James Jones (Rams) has scored 11.0 points.`

Pass/Fail: *Pass*