Professor Wumpus

By: Joseph Reidell and Luis Taylor

CS 1632 - DELIVERABLE 1: Test Plan and Traceability Matrix

Introduction

The following test plan attempts to give a comprehensive scope of the required functionalities of the Professor Wumpus game. There are test cases for each requirement and in most instances there are several test cases. The test cases are displayed first followed by the Traceability Matrix and a report of the defects found throughout the testing process.

We experienced more trouble than expected in testing the warning messages concerning the TA. The process was a bit involved due to the fact that the TA moves randomly. Furthermore, the positions change with each different seed. However, we wanted to test that the messages appeared under various different seeds and coming in from every direction.

We decided to highlight the certain directional test cases with Professor Wumpus. Professor Wumpus is meant to stay in the same one random room in any seed. That's why we decided to test that it was accurately portrayed from all different directions. We did these tests with Near Prof. Wumpus North Test, Near Prof. Wumpus South Test, Near Prof. Wumpus East Test, Near Prof. Wumpus West Test. We also decided to focus on the program's input validation. We did this with tests like the Invalid and Valid Seed tests along with others. We just wanted to make sure that the program did not fully break in these scenarios.

Our process for thinking of edge cases began with some of the implicit boundary values. That is why we decided to test the MAX_INT and MIN_INT as values for the game's seed. We do this in the Max 32-Bit Integer Test and Min 32-Bit Integer Test. Our other edge case comes from testing unexpected use. We passed in a non numeric value in for the seed in the Invalid Seed Test.

Test Plan

Identifier: Matrix Check
Description: A matrix grid of 6x6 is loaded up on the screen after compilation.
Preconditions: The program compiles and runs with some 32-bit integer seed entered in.
Execution steps: Run the program with the command java -jar profwumpus.jar and a 6x6 matrix is displayed on the screen.
Postconditions: The program runs and displays a 6x6 matrix with the students location in one of the matrices.

Identifier: Student Check
Description: The representation of the student(S) is displayed within one the matrices
Preconditions: The program compiles and runs with some 32-bit integer seed entered in.
Execution steps: Run the program with the command java -jar profwumpus.jar and a 6x6 matrix is displayed on the screen along with an ‘S’ inside of a block on the top left of the matrix.
Postconditions: The program runs and displays a 6x6 matrix with the students location in one of the matrices.

Identifier: Valid Direction
Description: Type in a E to go East and ensure the student moves to the right.
Preconditions: The program compiles and runs with some 32-bit integer seed entered in and asks the user what direction they want to move the student.
Execution Steps: Run the program and a 6x6 matrix is displayed on the screen along with an ‘S’ inside of a block. Then type in an E to move East and the student will move to a block on the right.
Postconditions: When the student moves, it should either move a block to the right or tell the user “There’s a wall there, buddy!”.

Identifier: Invalid Direction
Description: Type in a P as a direction and the program should tell the user "Please enter N, S, E, or W".
Preconditions: The program compiles and runs with some 32-bit integer seed entered in and asks the user what direction they want to move the student.
Execution Steps: Run the program and a 6x6 matrix displays along with the students location. Enter in a P instead of a valid direction like E to see if the message displays.
Postconditions: The student should not move and the program should display the message: "Please enter N, S, E, or W"

Identifier: Uppercase Letter Direction
Description: Type in an uppercase S to move the student south.
Preconditions: The program compiles and runs with some 32-bit integer seed entered in and asks the user what direction they want to move the student.

Execution Steps: Run the program and a 6x6 matrix displays along with the students location. Type in an uppercase S and the student should move down a block.
Postconditions: The student moves down a block.

Identifier: Lowercase Letter Direction
Description: Type in a lowercase s to move the student south.
Preconditions: The program compiles and runs with some 32-bit integer seed entered in and asks the user what direction they want to move the student.
Execution steps: Run the program and a 6x6 matrix displays along with the students location. Type in a lowercase s and the student should move down a block.
Postconditions: The student moves down a block.

Identifier: Existing Room Test
Description: This test verifies that when a student moves in the direction of the existing room and a new iteration begins
Preconditions: The program runs with 30 passed in as the seed and displays a 6x6 matrix with the student in the top left of the matrix
Execution steps: Enter in east as the direction
Postconditions: The 6x6 matrix is displayed and the student has moved into the existing room to the east. A new iteration begins

Identifier: Nonexistent Room Test
Description: This test verifies that if a student tries to move in the direction of a nonexistent room the game indicates the user that they cannot move in that direction
Preconditions: The program runs with 30 passed in as the seed and displays a 6x6 matrix with the student in the top left of the matrix
Execution steps: Enter in north as the direction
Postconditions: The message “There’s a wall there, buddy” is printed onto the screen

Identifier: Valid Seed Test
Description: Tests to see that a valid seed value is accepted by the program
Preconditions: the program complies and is being run on the command line with the seed as an argument
Execution steps: run the program with the command `java -jar profwumpus.jar 25`
Postconditions: The program runs and displays the seed value passed in as “Playing with seed: 25”

Identifier: Invalid Seed Test
Description: This is one of the edge cases tested for this test plan. Tests to see that an invalid seed value is ignored by the program
Preconditions: the program compiles and is being run on the command line with the seed as an argument
Execution steps: run the program with the command `java -jar profwumpus.jar 3222ffd`

Postconditions: The program runs and displays a random valid seed on the screen as “Playing with seed: *random value*”

Identifier: Win Test
Description: This test verifies that if a player has found the assignment and then encounters Professor Wumpus, they win the game. Then the program terminates
Preconditions: The program runs with 100 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions in the following order: east, east, east, south, south, south, south
Postconditions: Player wins and this message is displayed: “You turn in your assignment. YOU WIN!” and program terminates.

Identifier: Lose Test
Description: This test verifies that if a player has not found the assignment and then encounters Professor Wumpus, they lose the game. Then the program terminates
Preconditions: The program runs with 100 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions in the following order: south, south, south, south, east, east, east
Postconditions: Player loses and the following message is displayed: “Prof Wumpus sees you, but you don’t have your assignment. YOU LOSE!” and the program terminates.

Identifier: Near Prof. Wumpus North Test
Description: Approach the room Professor Wumpus is in from the south and hear him pontificating about computer science
Preconditions: The program runs with 4216 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the direction in the following order: south, south, south, east, east, east, east
Postconditions: The game will display “You hear someone pontificating on Computer Science... Professor Wumpus must be nearby!”. He is to the north one block.

Identifier: Near Prof. Wumpus South Test
Description: Approach the room Professor Wumpus is in from the north and hear him pontificating about computer science
Preconditions: The program runs with 3127 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the direction in the following order: east, east, east, south,
Postconditions: The game will display “You hear someone pontificating on Computer Science... Professor Wumpus must be nearby!”. He is to the south one block.

Identifier: Near Prof. Wumpus East Test

Description: Approach the room Professor Wumpus is in from the west and hear him pontificating about computer science
Preconditions: The program runs with 1256 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the direction in the following order: south, south, south, south
Postconditions: The game will display “You hear someone pontificating on Computer Science... Professor Wumpus must be nearby!”. He is to the east one block.

Identifier: Near Prof. Wumpus West Test
Description: Approach the room Professor Wumpus is in from the east and hear him pontificating about computer science
Preconditions: The program runs with 20 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions in the following order: east, east, south, south, south.
Postconditions: The game will display “You hear someone pontificating on Computer Science... Professor Wumpus must be nearby!”. He is to the west one block.

Identifier: Near TA West Test
Description: This test verifies that if the student is in the room directly to the west of the TA, the player will be notified
Preconditions: The program runs with 1436 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions as follows: east, east, east
Postconditions: The program prints the message: “You hear the shuffling of graded papers... the TA must be nearby!” The TA is to the east one block

Identifier: Near TA North Test
Description: This test verifies that if the student is in the room directly to the North of the TA, the player will be notified
Preconditions: The program runs with 222 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions as follows: east, east, east, east
Postconditions: The program prints the message: “You hear the shuffling of graded papers... the TA must be nearby!” The TA is to the south one block

Identifier: Near TA East Test
Description: This test verifies that if the student is in the room directly to the East of the TA, the player will be notified
Preconditions: The program runs with 333 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter in the directions as follows: south, south, south, south, east, east, east, east, north, north, north, north

Postconditions: The program prints the message: “You hear the shuffling of graded papers... the TA must be nearby!” The TA is to the south one block

Identifier: Near TA South Test
Description: This test verifies that if the student is in the room directly to the South of the TA, the player will be notified
Preconditions: The program runs with 444 passed in as the seed and displays a 6x6 matrix with the student starting in the top left of the matrix
Execution steps: Enter the directions as follows: east, east, east, east, south, south, south, west, west, west, west, south, north, south, north, north
Postconditions: The program prints the message: “You hear the shuffling of graded papers... the TA must be nearby!” The TA is to the north one block

Identifier: Max 32-Bit Integer Test
Description: This is one of the edge cases tested for this test plan. This tests how high the 32-bit seed integer can go before throwing an exception. The maximum integer for a 32-bit integer is 2,147,483,647.
Preconditions: The program compiles and runs with a 32-bit integer seed entered in
Execution steps: Enter into the command line, java -jar profwumpus.jar 2147483647
Postconditions: The program runs and displays the seed 2147483647

Identifier: Min 32-Bit Integer Test
Description: This is one of the edge cases tested for this test plan. This tests how high the 32-bit seed integer can go before throwing an exception. The minimum integer for a 32-bit integer is -2,147,483,647.
Preconditions: The program compiles and runs with a 32-bit integer seed entered in
Execution steps: Enter into the command line, java -jar profwumpus.jar -2147483647
Postconditions: The program runs and displays the seed -2147483647

Identifier: TA Wall Test
Description: This tests whether the program informs the student that the TA tried to go into a nonexistent room and they hear the TA bump into a wall.
Preconditions: Run and compile the program by using the command line java -jar profwumpus.jar 452112424
Execution steps: Follow the directions as follows: south
Postconditions: The message “You hear a thud, as if the TA hit into a Southern wall…” will display, indicating that the TA bumped into a wall.

Traceability Matrix

Matrix-Req: Matrix Check, Student Check
Direction-Req: Valid Direction, Invalid Direction
Case-sensitivity-Req: Uppercase Letter Direction, Lowercase Letter Direction
Room Exists-Req: Existing Room Test
Room Doesn't Exist-Req: Nonexistent Room Test
Valid Seed-Req: Valid Seed Test
Invalid Seed-Req: Invalid Seed Test, Max 32-Bit Integer Test, Min 32-Bit Integer Test
TA Wall-Req: TA Wall Test
Win/Lose-Req: Win Test, Lose Test
WumpusMessage-Req: Near Prof. Wumpus North Test, Near Prof. Wumpus South Test, Near Prof. Wumpus East Test, Near Prof. Wumpus West Test
TA Message-Req: Near TA West Test, Near TA North Test, Near TA East Test, Near TA South Test

Defects

SUMMARY: ArrayIndexOutofBounds exception at any Eastern wall.
DESCRIPTION: The test case associated with this defect is Nonexistent Room Test. Use any seed that you want to enter into the program and head towards the east wall. Once you get to the east wall, try to hit into the wall. If you do, it will cause an ArrayIndexOutofBounds exception and crash the program.
REPRODUCTION STEPS:

1. Start the program with any number seed and follow these directions: east, east, east, east
2. You will be right at the top right corner of the matrix. Go east once more
3. There will an ArrayIndexOutofBounds exception and the game terminates

EXPECTED BEHAVIOR: When you hit into any wall, whether it is north, south, east, or west, the message “There’s a wall there, buddy” should display.
OBSERVED BEHAVIOR: When you hit any part of the east wall, the compiler throws an ArrayIndexOutofBounds exception and crashes the program.

SUMMARY: Invalid seed does not get ignored.
DESCRIPTION: The test case associated with this defect is Invalid Seed Test. When you enter in your own seed number and add in letters or characters, it will start the game but it throws a NumberFormatException before it loads up the matrix.
REPRODUCTION STEPS:

1. On the command line, type in java -jar profwumpus.jar 452e@2
2. The title Professor Wumpus will pop up then a NumberFormatException error pops up, terminating the game entirely.

EXPECTED BEHAVIOR: When a invalid seed is typed in, the program should ignore that seed and randomize a different seed.
OBSERVED BEHAVIOR: When an invalid seed is typed in, the program starts but when it comes to displaying the seed number, it throws a NumberFormatException and terminates the game.

SUMMARY: TA and Professor Wumpus are in the same room.
DESCRIPTION: The test case associated with this defect is Lose Test. When Professor Wumpus is found, if you have the assignment or not, you should win or lose the game. However, if the TA is in the same room as Professor Wumpus, the student flees the room and you don’t win or lose the game. The TA takes priority over Professor Wumpus.
REPRODUCTION STEPS:

1. On the command line, type in java -jar profwumpus.jar 8673101
2. Follow these directions: south, south, south, east, east
3. The messages “You smell the ink of a printed-out assignment!” “You hear the shuffling of graded papers... the TA must be nearby!” “You hear someone pontificating on Computer Science... Professor Wumpus must be nearby!” should display

4. Go east one more time and the message “You see the TA and flee in terror to a random room!” should display and you run away to the top left corner
5. Follow the same exact directions as in step 2 except go east one more time
6. Professor Wumpus will be in the same room the TA was in and you will lose the game.

EXPECTED BEHAVIOR: Whether you found the assignment or not, once you find Professor Wumpus, the game should tell you that you have won or lost the game.
OBSERVED BEHAVIOR: The TA and Professor Wumpus are in the same room. Once you enter that room the TA makes you flee away instead of you winning or losing the game.

SUMMARY: The matrix displayed is 5x5 not 6x6.
DESCRIPTION: The test case associated with this defect is Matrix Check. The game should display a 6x6 matrix for the game. Instead it displays a 5x5 matrix.
REPRODUCTION STEPS:

1. On the command line, type in java -jar profwumpus.jar

EXPECTED BEHAVIOR: The program displays a 6x6 matrix on the screen
OBSERVED BEHAVIOR: The program displays a 5x5 matrix on the screen