**PROFESSOR WUMPUS**

CS 1632 – Deliverable 1:

Test Plan and Traceability Matrix

Group Members:

Anthony Poerio ([adp59@pitt.edu](mailto:adp59@pitt.edu))

Brandon Hedges ([bjh86@pitt.edu](mailto:bjh86@pitt.edu))

**INTRODUCTION**

Our chief concern while testing Profesor Wumpus was that we would *only* be able to do black-box testing, given that we did not have access to any of the source code, or knowledge of the program’s internals. Despite this concern, black-box testing a game like Professor Wumpus proved beneficial, because—as testers—we were able to experience the game exactly as end users would. And so, we were able to find defects naturally, in many of the same situations a real client would, rather than constructed cases based on our own knowledge of the code base.

Put another way: testing Professor Wumpus without any knowledge of the how it was made was a challenging, but useful, exercise because we were not incentivized to avoid—or attack—any particular areas of the requirements based on our knowledge of the system. The process was much more natural.

Having said that, testing Professor Wumpus was a mixed bag, with requirements that ranged from mildly to very difficult to test. Moreover, black-box testing was especially challenging for several of the game’s requirements that depended on either **a)** randomness, or **b)** knowledge of where game characters or items may be at any given time. Thus, while black-box testing Professor Wumpus, had necessarily relied only on what was printed to the console, which itself could be inaccurate (I.e. – itself subject to defects).

One of the most difficult requirements to test was that the print statement indicated that the TA was would be printed when the student was North, South, East, or West of the TA. This is difficult because the only indicator the user has of the location of the TA is the string that the TA is nearby. So in order to test whether the string printed, the movements of the TA with a specific input for the seed had to be mapped by intentionally running into the TA and losing the game many times. This had to be done before it could be reliably known that the TA was actually North, South, East or West of the student and the test case could be written.

Another challenging test was writing the execution steps for the testing the wall to wall boundaries. Part of writing these execution steps is preparing an exhaustive route that can test all the walls and avoid the TA in the process. Once again the TA’s position must be mapped before the test can begin and an appropriate seed must be found that allows the user to visit all the walls without interruption. This is a long and tedious process during black box testing.

Black box testing would be easier if there was some way to know where the Professor and the TA are, as a tester, otherwise it’s impossible to be 100% certain that the TA moves randomly, for example. Randomness needs to be statistically proven. Not definitively knowing all the TA’s moves makes it impossible to verify via black box testing. Assumptions can only be made based on what can be reasonably tested. White box testing is a start and has its place, but a full test plan should be a combination of white or grey box to be sure that testing is optimally accurate and the requirements are met.

Testing edge-cases for Professor Wumpus was comparatively simple, next to testing the elements that relied on some degree of randomness. For edge-cases, we focused on the requirements which specified user-input, and then tested values right at either end of the acceptable range. This proved fruitful, and we were able to find a defect using this particular method.

All in all, we are thankful for the opportunity to test a program like Professor Wumpus, under these circumstances. It has illustrated the positives and negatives of black-box testing, and provided a general understanding of the testing process, and why manual testing is not especially desirable—it’s tedious! We believe that with automated testing, all of the same actions could be done much more efficiently, and so that’s the chief area of improvement we would select as QA analysts for improvement to, and expansion of, this test suite.

**TEST CASES**

CASE 1**:**

IDENTIFIER: TEST-DISPLAY-ON-WUMPUS-INTERCEPT

TEST CASE: Run the program and intentionally walk into Prof Wumpus’s room to ensure the matrix always displays user’s location at each iteration—even when we are intercepted by Prof Wumpus.

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1. With seed 1, Professor Wumpus will be placed in the room at ROW=1, COL=4 (indexing from 1).

EXECUTION STEPS: Move EAST 3 times. Press: E, Enter; E, Enter; E, Enter;

POSTCONDITIONS: First: The user’s location is displayed in the matrix at ROW=1, COL=4. Second: Display the message “Prof Wumpus sees you, but you don't have your assignment. YOU LOSE!” Third: Exit program.

CASE 2**:**

IDENTIFIER: TEST-MATRIX-DIMENSIONS

TEST CASE: Ensure the program displays a matrix with dimensions 6x6 (6 rows, 6 columns)

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS: Count the number of COLUMNS in the matrix. Count the number of columns in the matrix, starting at 1.

POSTCONDITIONS: You will finish counting at the number 6.

CASE 3**:**

IDENTIFIER: TEST-INVALID-INPUT-LETTER

TEST CASE: Run the program with invalid letters input parameters

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following invalid inputs:

1) “o”

2) “p”

1 3) “pq”

4) “north”

POSTCONDITIONS: For each invalid input, the Student marker will not move, and the program will output: “Please enter N, S, E, or W”

CASE 4**:**

IDENTIFIER: TEST-INVALID-INPUT-NUMBER

TEST CASE: Run the program with invalid numbers as input parameters

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following invalid inputs:

1) “1”

2) “-2”

3) “10”

4) “1000000000”

POSTCONDITIONS: For each invalid input, the Student marker will not move, and the program will output: “Please enter N, S, E, or W”

CASE 5**:**

IDENTIFIER: TEST-VALID-LOWERCASE-INPUT

TEST CASE: Run the program and pass in N,E,S,and W in lowercase form

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following invalid inputs:

1) “e”

2) “s”

3) “w”

4) “n”

POSTCONDITIONS: For each input, the student marker will move in the direction specified.

CASE 6**:**

IDENTIFIER: TEST-VALID-LOWERCASE-INPUT-AT-WALLS

TEST CASE: Test that valid lowercase input is accepted when the move itself is invalid because the user would be walking into a wall

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following inputs:

1) “n” 🡪 1x

2) “s” 🡪 5x (until bottom left wall is reached)

3) “w” 🡪 1x

4) “e” 🡪 5x (until bottom right wall is reached)

POSTCONDITIONS:

Each time the user attempts to walk into a wall—the program will accept the lowercase input, and respond with the following error message: “There’s a wall there, buddy!”

CASE 7**:**

IDENTIFIER: TEST-ALL-VALID-ROOMS-WITH-UPPERCASE-INPUT

TEST CASE: Run the program and move through the matrix into each valid room, ensuring that the user can enter into all rooms that are valid.

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following inputs:

1) “S” 🡪 4x

2) “E” 🡪 1x

3) “N” 🡪 4x

4) “E” 🡪 1x (will encounter TA and move to room at ROW=5, COL=3)

5) “N” 🡪 4x (Wumpus will be 1 room to right, so need to avoid him)

6) “S” 🡪 1x

7) “E” 🡪 1x

8) “S” 🡪 3x

9) “E” 🡪 1x (Find Assignment)

10) “N” 🡪 4x

11) “W” 🡪 1x (Encounter Wumpus)

POSTCONDITIONS:

All moves will be valid, and we will see the S marker move into the room specified. The only time the S marker will NOT move into the room specified is when we encounter the TA at STEP 4, and we are moved into a random room instead. After step 11, the S marker will be in room: ROW=1, COL=4, and we will receive the message: “You turn in your assignment. YOU WIN.” The game will then end.

CASE 8**:**

IDENTIFIER: TEST-ALL-VALID-ROOMS-WITH-LOWERCASE-INPUT

TEST CASE: Run the program and move through the matrix into each valid room, ensuring that the user can enter into all rooms that are valid when lowercase input is used.

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 1

EXECUTION STEPS:

Pass in the following inputs:

1) “s” 🡪 4x

2) “e” 🡪 1x

3) “n” 🡪 4x

4) “e” 🡪 1x (will encounter TA and move to room at ROW=5, COL=3)

5) “n” 🡪 4x (Wumpus will be 1 room to right, so need to avoid him)

6) “s” 🡪 1x

7) “e” 🡪 1x

8) “s” 🡪 3x

9) “e” 🡪 1x (Find Assignment)

10) “n” 🡪 4x

11) “w” 🡪 1x (Encounter Wumpus)

POSTCONDITIONS:

All moves will be valid, and we will see the S marker move into the room specified. The only time the S marker will NOT move into the room specified is when we encounter the TA at STEP 4, and we are moved into a random room instead. After step 11, the S marker will be in room: ROW=1, COL=4, and we will receive the message: “You turn in your assignment. YOU WIN.” The game will then end.

CASE 9**:**

IDENTIFIER: TEST-NORTH-AND-SOUTH-WALLS

TEST CASE: Run the program and intentionally walk into the NORTH and SOUTH walls, ensuring that program indicates to the user that she cannot move outside the bounds of the game-space.

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 5

EXECUTION STEPS:

Pass in the following inputs:

1) “n” 🡪 1x (receive error message)

2) “e” 🡪 1x

3) “n” 🡪 1x (receive error message)

4) “e” 🡪 1x

5) “n” 🡪 1x (receive error message)

7) “e” 🡪 1x

8) “n” 🡪 1x (receive error message)

9) “e” 🡪 1x

10) “n” 🡪 1x (receive error message)

11) “s” 🡪 5x (receive error message on 5th s)

12) “w” 🡪 1x

11) “s” 🡪 1x (receive error message)

12) “w” 🡪 1x

13) “s” 🡪 1x (receive error message)

14) “w” 🡪 1x

13) “s” 🡪 1x (receive error message)

14) “w” 🡪 1x

15) “s” 🡪 1x (receive error message)

POSTCONDITIONS:

Each time the user walks into a wall, the program will output the message “There’s a wall there, buddy!”

CASE 10**:**

IDENTIFIER: TEST-EAST-AND-WEST-WALLS

TEST CASE: Run the program and intentionally walk into the EAST and WEST walls, ensuring that program indicates to the user that she cannot move outside the bounds of the game-space.

PRECONDITIONS: Start the program with the command: java -jar profwumpus.jar 5

EXECUTION STEPS:

Pass in the following inputs:

1) “w” 🡪 1x (receive error message)

2) “s” 🡪 1x

3) “w” 🡪 1x (receive error message)

4) “s” 🡪 1x

5) “w” 🡪 1x (receive error message)

7) “s” 🡪 1x

8) “w” 🡪 1x (receive error message)

9) “s” 🡪 1x

10) “w” 🡪 1x (receive error message)

11) “e” 🡪 5x (receive error message on 5th e)

12) “n” 🡪 1x

11) “e” 🡪 1x (receive error message)

12) “n” 🡪 1x

13) “e” 🡪 1x (receive error message)

14) “n” 🡪 1x

13) “e” 🡪 1x (receive error message)

14) “n” 🡪 1x

15) “e” 🡪 1x (receive error message)

POSTCONDITIONS:

Each time the user walks into a wall, the program will output the message “There’s a wall there, buddy!”

CASE 11**: (EDGE CASE)**

IDENTIFIER: TEST-RANDOM-NUMBER-GENERATOR-WITH-32-BIT-INTEGER

TEST CASE: Run the program and pass it an integer that is >= 32-bits long or less

PRECONDITIONS: None

EXECUTION STEPS: java -jar profwumpus.jar 2147483647 (Highest possible int)

\* This number is (2^31-1)

\* So it can be represented in less than 32-bits, signed

POSTCONDITIONS: The program starts as expected, and the game-space matrix is displayed. The game will output the message “Welcome to Professor Wumpus

Playing with seed 2147483647”

CASE 12**: (EDGE CASE)**

IDENTIFIER: TEST-RANDOM-NUMBER-GENERATOR-WITH-NEGATIVE-32-BIT-INTEGER

TEST CASE: Run the program and pass it a NEGATIVE integer that is >= 32-bits long or less

PRECONDITIONS: None

EXECUTION STEPS: java -jar profwumpus.jar -2147483647 (Lowest possible int)

\* This number is ( -(2^31) - 1 )

\* So it can be represented in less than 32-bits, signed

POSTCONDITIONS: The program starts as expected, and the game-space matrix is displayed. The game will output the message “Welcome to Professor Wumpus

Playing with seed -2147483647”

CASE 13**: (EDGE CASE)**

IDENTIFIER: TEST-RUNNING-WITH-SEED-BEYOND-BOUNDS

TEST CASE: Run the program and pass it an invalid seed that is not in the range of 32-bit signed integers.

PRECONDITIONS: Run the program with the command:

java -jar profwumpus.jar 2147483648 (Seed one greater than the range of 32-bit signed integers)

EXECUTION STEPS: java -jar profwumpus.jar 2147483648 (One over highest possible int)

POSTCONDITIONS: The program starts as expected, and the game-space matrix is displayed. The game will output the message “Welcome to Professor Wumpus

Playing with seed” (random seed displayed on each iteration).

CASE 14**:**

IDENTIFIER: TEST-RUNNING-WITH-FLOATING-POINT-SEED

TEST CASE: Run the program and pass it an invalid seed that is a floating point integer and not a seed.

PRECONDITIONS: Run the program with the command:

java -jar profwumpus.jar 0.5 (Seed with floating point number)

EXECUTION STEPS: java -jar profwumpus.jar 0.5

POSTCONDITIONS: The program starts as expected, and the game-space matrix is displayed. The game will output the message “Welcome to Professor Wumpus

Playing with seed” (random seed displayed on each iteration).

CASE 15**:**

IDENTIFIER: TEST-RANDOMNESS-OF-TA-MOVEMENTS

TEST CASE: Run the program 3 times and record the output. Compare the outputs for the TA’S movements against the other outputs to ensure that the output is not the same or too similar.

PRECONDITIONS: Run the program with the command: java -jar profwumpus.jar

EXECUTION STEPS:

1)Run program with command: java -jar profwumpus.jar

2) “s”

3) “n”

4) “s”

5) “n”

6) “s”

7) “n”

8) “s”

9) “n”

10) “s”

12) Record output from console

13) Ctrl-c to terminate program

14) Run program with command: java -jar profwumpus.jar

15) “s”

16) “n”

17) “s”

18) “n”

19) “s”

20) “n”

21) “s”

22) “n”

23) “s”

24) Record output from console

25) Ctrl-c to terminate program

26) Run program with command: java -jar profwumpus.jar

27) “s”

28) “n”

29) “s”

30) “n”

31) “s”

32) “n”

33) “s”

34) “n”

35) “s”

36) Record output from console

37) Ctrl-c to terminate program

38) Compare output data

POSTCONDITIONS: In each iteration of the program, the TA bumps into random walls in random orders and the output of the TA’S movements from each instance of the program do not display the same TA movements.

CASE 16**:**

IDENTIFIER: TEST-DOES-WUMPUS-MOVE

TEST CASE: Seed the program with a specific number to ensure Professor Wumpus’s starting point is known. Then move to the rooms around professor Wumpus and move around the perimeter of his room.

PRECONDITIONS: java -jar profwumpus.jar 1

EXECUTION STEPS:

1) java -jar profwumpus.jar 1

2)”e”

3)”e” Hear Wumpus

4)”s”

5)”e” Hear Wumpus

6)”e”

7)”n” Hear Wumpus

8)”w” Encounter Wumpus

POSTCONDITIONS: Wumpus will be heard every time the student is next to Row 1 column 4 and encountered when the student moves to that room.

CASE 17**:**

IDENTIFIER: TEST-WIN-IF-HAVE-ASSIGNMENT-AND-ENCOUNTER-WUMPUS

TEST CASE: Run the program. Find the assignment then return to Wumpus.

PRECONDITIONS: java -jar profwumpus.jar 1

EXECUTION STEPS:

1) java -jar profwumpus.jar 1

2) “s”

3) “s”

4) “s”

5) “s”

6) “e”

7) “e”

8) “e”

9) “e” (Find assignment)

10) “n”

11) “n”

12) “n”

13) “n”

14) ”w” (Encounter Wumpus)

POSTCONDITIONS: When the assignment is found and Professor Wumpus is encountered the player will win the game and the program will terminate.

CASE 18**:**

IDENTIFIER: TEST-LOSE-IF-DO-NOT-HAVE-ASSIGNMENT-AND-ENCOUNTER-WUMPUS

TEST CASE: Start program with one as the seed and move directly to Professor Wumpus.

PRECONDITIONS: java -jar profwumpus.jar 1

EXECUTION STEPS:

1) java -jar profwumpus.jar 1

2) “e”

3) “e”

4) “e” (Encounter Wumpus)

POSTCONDITIONS: When Wumpus is encountered when the player does not have the assignment the player will lose and the game will terminate.

CASE 19**:**

IDENTIFIER: TEST-NORTH-SOUTH-EAST-WEST-PONTIFICATION

TEST CASE: Run the program and move to the rooms directly North, South ,East and west of Wumpus.

PRECONDITIONS: java -jar profWumpus.jar 2

EXECUTION STEPS:

1. java -jar profWumpus.jar 2
2. “e”
3. “e”
4. “s”
5. “s” (Hear Wumpus from the North)
6. “e”
7. “s” (Hear Wumpus from the East)
8. “s”
9. “w” (Hear Wumpus from the South)
10. “w”
11. “n” (Hear Wumpus from the West)
12. Terminate program

POSTCONDITIONS: The program will start up normally and when the player moves to a location north,south,east or west of Wumpus, a string will be displayed to the user stating that Wumpus is pontificating nearby.

CASE 20**:**

IDENTIFIER: TEST-NO-INVALID-PONTIFICATION

TEST CASE: Run the program and move through all of the rooms that are not north, south east or west of Wumpus.

PRECONDITIONS: java -jar profWumpus.jar 3

EXECUTION STEPS:

1. java -jar profWumpus.jar 3
2. “s”
3. “s”
4. “e”
5. “e”
6. “s”
7. “s”
8. “e”
9. “n”
10. “n”
11. ”n”
12. “n”
13. “e”
14. “s”
15. “s”
16. “s”
17. “s”
18. “w”
19. “w”
20. “n”
21. “n”
22. “n”
23. “n”
24. “w”
25. “s”
26. Terminate program

POSTCONDITIONS: Pontification is not heard by the student at all at any time during this sequence.

CASE 21**:**

IDENTIFIER: TEST-TA-NORTH-EAST-DETECTION

TEST CASE: Run the program with the command: java -jar profWumpus.jar 3. Move to positions east and north of the TA.

PRECONDITIONS: java -jar profWumpus.jar 3

EXECUTION STEPS:

1. java -jar profWumpus.jar 3
2. “s” (TA is east of student)
3. (TA is north of student)
4. Terminate program

POSTCONDITIONS: When the TA is East or North of the student a message will be displayed that the TA is nearby.

CASE 22**:**

IDENTIFIER: TEST-TA-SOUTH-WEST-DETECTION

TEST CASE: Run the program with the command: java -jar profWumpus.jar 3. Move to positions east and south of the TA.

PRECONDITIONS: java -jar profWumpus.jar 35

EXECUTION STEPS:

1. java -jar profWumpus.jar 35
2. (TA is south of student)
3. “e”
4. “s” (TA is west of student)
5. Terminate program

POSTCONDITIONS: When the TA is South or West of the student a message will be displayed that the TA is nearby.

**TRACEABILITY MATRIX**

|  |  |  |
| --- | --- | --- |
| TEST CASE | REQUIREMENT | PASS/FAIL |
| Case 1: TEST-DISPLAY-ON-WUMPUS-INTERCEPT | 1. The game shall consist of a 6 by 6 matrix of rooms, which shall be displayed to the player at each iteration, along with the location of the Student (indicated as an S). | **FAIL. DEFECT-1** |
| Case 2: TEST-MATRIX-DIMSIONS | 1. The game shall consist of a 6 by 6 matrix of rooms, which shall be displayed to the player at each iteration, along with the location of the Student (indicated as an S). | **FAIL. DEFECT-2** |
| Case 3: TEST-INVALID-INPUT-LETTER | 2. At each iteration, the player shall be able to input the direction they wish the Student to move (N for North, S for South, E for East or W for West). There are no other options. If a player enters any command other than N, S, E, or W, or their lowercase equivalents, the game shall display "Please enter N, S, E, or W". | **PASS** |
| Case 4: TEST-INVALID-INPUT-NUMBER | 2. At each iteration, the player shall be able to input the direction they wish the Student to move (N for North, S for South, E for East or W for West). There are no other options. If a player enters any command other than N, S, E, or W, or their lowercase equivalents, the game shall display "Please enter N, S, E, or W". | **PASS** |
| Case 5: TEST-VALID-LOWERCASE-INPUT | 3. All user input shall be case-insensitive. For example, typing either "N" or "n" shall take the Student to the room to the North (if it exists). | **PASS** |
| Case 6: TEST-VALID-LOWERCASE-AT-WALLS | **3. All user input shall be case-insensitive. For example, typing either "N" or "n" shall take the Student to the room to the North (if it exists).** | **FAIL. DEFECT-3** |
| Case 7: TEST-ALL-VALID-ROOMS-WITH-UPPERCASE-INPUT | 4. If a room exists in the direction they have indicated, the Student shall move to that room and a new iteration shall commence. | **FAIL. DEFECT-1** |
| Case 8: TEST-ALL-VALID-ROOMS-WITH-UPPERCASE-INPUT | 4. If a room exists in the direction they have indicated, the Student shall move to that room and a new iteration shall commence. | **FAIL. DEFECT-1** |
| Case 9: TEST-NORTH-AND-SOUTH-WALLS | 5. If a room does not exist, the game shall indicate to the user that they cannot move in that direction. | **PASS** |
| Case 10: TEST-EAST-AND-WEST-WALLS | 5. If a room does not exist, the game shall indicate to the user that they cannot move in that direction. | **FAIL. DEFECT-3** |
| Case 11: TEST-RANDOM-NUMBER-GENERATOR-WITH-32-BIT-INTEGER | 6. The game shall accept a 32-bit signed integer seed for the random number generator. This should be entered as an argument for the program on the command line. | **PASS** |
| Case 12: TEST-RANDOM-NUMBER-GENERATOR-WITH-NEGATIVE-32-BIT-INTEGER | 6. The game shall accept a 32-bit signed integer seed for the random number generator. This should be entered as an argument for the program on the command line. | **PASS** |
| Case 13: TEST-RUNNING-WITH-SEED-BEYOND-BOUNDS | **7. If an invalid value is entered for the seed, the system shall ignore it and assume that no argument was passed in (that is, it will act as though no seed were entered).** | **FAIL. DEFECT-4** |
| Case 14:TEST-RUNNING-WITH-FLOATING-POINT-SEED | **7. If an invalid value is entered for the seed, the system shall ignore it and assume that no argument was passed in (that is, it will act as though no seed were entered).** | **FAIL. DEFECT-4** |
| Case 15:TEST-RANDOMNESS-OF-TA-MOVEMENTS | **8. The TA shall move a random direction at each iteration. If the TA attempts to move to a room which does not exist, the user shall be informed that they hear the TA bump into a wall. Professor Wumpus shall not move.** | **PASS** |
| Case 16:TEST-DOES-WUMPUS-MOVE | **8. The TA shall move a random direction at each iteration. If the TA attempts to move to a room which does not exist, the user shall be informed that they hear the TA bump into a wall. Professor Wumpus shall not move.** | **PASS** |
| Case 17: TEST-WIN-IF-HAVE-ASSIGNMENT-AND-ENCOUNTER-WUMPUS | **9. If the Student has found the Assignment and encountered Professor Wumpus, the player shall win. If the Student has encountered Professor Wumpus but has not found the Assignment, the player shall lose. In either case, after the scenario occurs, the program shall end.** | **PASS** |
| Case 18: TEST-LOSE-IF-NOT-HAVE-ASSIGNMENT-AND-ENCOUNTER-WUMPUS | **9. If the Student has found the Assignment and encountered Professor Wumpus, the player shall win. If the Student has encountered Professor Wumpus but has not found the Assignment, the player shall lose. In either case, after the scenario occurs, the program shall end.** | **PASS** |
| Case 19: TEST-NORTH-SOUTH-EAST-WEST-PONTIFICATION | **10. If the Student is in a room directly to the North, South, East or West of Professor Wumpus, the program shall indicate that they hear somebody pontificating on Computer Science.** | **PASS** |
| Case 20: TEST-NO-INVALID-PONTIFICATION | **11. If the Student is in a room directly to the North, South, East or West of the TA, the program shall indicate that they hear the rustling of graded papers.** | **PASS** |
| Case 21: TEST-TA-NORTH-EAST-DETECTION | **11. If the Student is in a room directly to the North, South, East or West of the TA, the program shall indicate that they hear the rustling of graded papers.** | **PASS** |
| Case 22: TEST-TA-SOUTH-WEST-DETECTION | **11. If the Student is in a room directly to the North, South, East or West of the TA, the program shall indicate that they hear the rustling of graded papers.** | **PASS** |

**DEFECTS FOUND**

DEFECT 1**:**

SUMMARY: Matrix Display NOT updated when Student is intercepted by Prof Wumpus

DESCRIPTION: When a user walks into Professor Wumpus, the location is not updated before ending the game.

**CASES THAT CAUSED DEFECT: Case 1,7 and 8**

REPRODUCTION STEPS: 1) Start the program with the command: java -jar profwumpus.jar 1. 2) Move EAST 4 times. Press: E, Enter; E, Enter; E, Enter; E, Enter.

EXPECTED BEHAVIOR: First: The user’s location is displayed in the matrix at ROW=1, COL=4. Second: Display the message “Prof Wumpus sees you, but you don't have your assignment. YOU LOSE!” Third: Exit program.

OBSERVED BEHAVIOR: The user’s location is NOT updated, and Student remains in the matrix at ROW=1, COL=3—even though s/he has moved into room [1,4]. The correct message displays, and program exits.

DEFECT 2**:**

SUMMARY: Matrix is not 6x6

DESCRIPTION: The Matrix has dimensions 5x5, but the requirements specify that dimensions must be 6x6

**CASES THAT CAUSED DEFECT: Case 2**

REPRODUCTION STEPS: 1) Start the program with the command: java -jar profwumpus.jar 1. 2) Count the number of rows and columns that appear on the screen, starting at 1.

EXPECTED BEHAVIOR: There will be 6 rows and 6 columns.

OBSERVED BEHAVIOR: There are 5 rows and 5 columns

DEFECT 3**:**

SUMMARY: Program crashes when trying to walk into bottom right EAST wall

DESCRIPTION: The program crashes when the user tries to walk into the EAST side of the bottom right wall of the game-space matrix, using the lowercase input character “e”.

**CASES THAT CAUSED DEFECT: Case 6 and 10**

REPRODUCTION STEPS:

Start the program with the command: java -jar profwumpus.jar 1.

Pass in the following inputs:

1) “n” 🡪 1x

2) “s” 🡪 5x (until bottom left wall is reached)

3) “w” 🡪 1x

4) “e” 🡪 5x (until bottom right wall is reached)

EXPECTED BEHAVIOR: Each time the user attempts to walk into a wall—the program will accept the lowercase input, and respond with the following error message: “That’s a wall there, buddy!”

OBSERVED BEHAVIOR: When the user attempts to walk into the bottom right wall, using the input “e”, the program crashes and outputs the following error message:

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5

at ProfWumpus.moveStudent(ProfWumpus.java:51)

at ProfWumpus.playGame(ProfWumpus.java:335)

at ProfWumpus.main(ProfWumpus.java:362)

DEFECT 4**:**

SUMMARY: Program crashes when an invalid seed is passed in as an argument.

DESCRIPTION: The requirements specify that if an invalid seed is entered that it should be ignored and the program should run normally.

**CASES THAT CAUSED DEFECT: Case 13 and 14**

REPRODUCTION STEPS: 1) Start the program with the command: java -jar profwumpus.jar 2147483648 OR java -jar profwumpus.jar 0.5

EXPECTED BEHAVIOR: The seed will be ignored and the program will run normally.

OBSERVED BEHAVIOR: When 2147483648 is entered as the seed the program prints “Welcome to Professor Wumpus” then throws an exception and prints the stack trace “Exception in thread "main" java.lang.NumberFormatException: For input string: "2147483648"

at java.lang.NumberFormatException.forInputString(Unknown Source)

at java.lang.Integer.parseInt(Unknown Source)

at java.lang.Integer.parseInt(Unknown Source)

at ProfWumpus.main(ProfWumpus.java:358)”

When 0.5 is entered as the seed the program prints “Welcome to Professor Wumpus” then throws an exception and prints the stack trace “Exception in thread "main" java.lang.NumberFormatException: For input string: "0.5"

at java.lang.NumberFormatException.forInputString(Unknown Source)

at java.lang.Integer.parseInt(Unknown Source)

at java.lang.Integer.parseInt(Unknown Source)

at ProfWumpus.main(ProfWumpus.java:358)”