University of Pittsburgh

Professor Wumpus

CS1632 - Deliverable 1: Test Plan and Traceability Matrix

Audrey Ho and Sam Nigh

Bill Laboon

1/30/2018

TABLE OF CONTENTS

1. INTRODUCTION

Professor Wumpus is a simple, terminal-based game where the objective is for the Student to find their assignment and turn it in to Professor Wumpus. The Student must travel to different rooms to find their assignment before meeting Professor Wumpus. If the Student encounters the TA, who also wanders the rooms, the Student will flee into a random room; however, if the Student finds Professor Wumpus before finding their assignment, the player loses. The player wins if they encounter Professor Wumpus after finding their assignment.

The purpose of this deliverable is to outline the test plan for *Professor Wumpus* based on the requirements, and to demonstrate our thought process when finding defects in the game.

The next pages list the test cases for each requirement. Most requirements have more than one test case. There are also sections for the traceability matrix and defects found. Some thought processes included validation of input from the user before and during the game, responding to user actions without failure of the system, game success and failure requirements, and path-based testing such as encountering Professor Wumpus before finding the assignment (failure) and after finding the assignment (success). As an example edge case, we tested seed validation to detect that it would ignore a seed argument that is greater than maximum integer value or less than minimum integer value in Java. This is to determine whether the game would work when an invalid integer is entered.

As a note, we are indexing the 6x6 room game map as **starting from 0**, and from the **top left corner**. For example: [0][0] is the top left room on the map. Overall, we argue that we have the appropriate test plan for *Professor Wumpus*, outlined in the following document.

2. TEST CASES

IDENTIFIER: 0

DESCRIPTION: Enter a number greater than the largest 32-bit signed integer as a command-line argument. This to ensure the program detects an invalid integer value, ignoring the command line argument and generating its own seed.

PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar 2147483648** in the command prompt/terminal.

POSTCONDITIONS: The program generates a random seed, ignoring the command-line argument.

IDENTIFIER: 1

DESCRIPTION: Enter a number 1 less than the smallest 32-bit signed integer as a command-line argument. This is to ensure the program detects an invalid integer value, ignoring the command line argument and generating its own seed.

PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar -2147483649** into the command prompt/terminal

POSTCONDITIONS: The program generates a random seed, ignoring the command-line argument.

IDENTIFIER: 2

DESCRIPTION: Enter a string as the command-line argument. This is to ensure the program detects a non-integer value (string object), ignoring the command line argument and generating its own seed.

PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar howdy** (or another arbitrary string argument) into the command prompt terminal.

POSTCONDITIONS: The program generates a random seed, ignoring the command-line argument.

IDENTIFIER: 3

DESCRIPTION: Enter a character as the command-line argument. This is to ensure the program detects a non-integer value (string object or character type), ignoring the command line argument and generating its own seed.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar s** into the command prompt terminal.

POSTCONDITIONS: The program generates a random seed, ignoring the command-line argument.

IDENTIFIER: 4

DESCRIPTION: This is to test that the TA will move randomly each turn.
PRECONDITIONS:

- A terminal or command prompt window is open
- The current working directory is the path of the **profwumpus.jar** file

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar 100** into the command line terminal.
2) Navigate to point [4,3], 4th row and 3rd column by following the path: east, east, east, south, south, south, north. The game will print "You hear the shuffling of graded papers… the TA must be nearby!"
3) Restart the game except using the command **java -jar profwumpus.jar 10** and follow the path described in step 2.

POSTCONDITIONS: The game does not print the message "You hear the shuffling of graded papers … the TA must be nearby!" because the TA is moving based on a different seed.

IDENTIFIER: 5

DESCRIPTION: To test that for each turn, if the TA tries to move into a room that does not exist, the TA will hit a wall.
PRECONDITIONS: A terminal or command prompt window is open

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate to point [1,2], the 2nd row and 3rd column using the path: south, south, east, south, north. The game will print "You hear a thud, as if the TA hit into a Northern wall".

POSTCONDITIONS: The TA does not go past the northern wall and remains in the game's bounds.

IDENTIFIER: 6

DESCRIPTION: For each turn, Professor Wumpus shall not move.
PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate to the point [4, 2] 5th row, 3rd column using the path: south, south, south, south, east, east. The game will print, “You hear someone pontificating on Computer Science… Professor Wumpus must be nearby!”
2) Move north once. The game will not print any statements about Professor Wumpus (Professor Wumpus is at point [4, 3]).
3) Move east once. The game will print “You hear someone pontificating on Computer Science … Professor Wumpus must be nearby!”.

POSTCONDITIONS: After the game first prints that Professor Wumpus is nearby (after step 1) and the Student goes north, the game no longer prints any statements about Professor Wumpus. However, when moving east after will indicate that Professor Wumpus is nearby showing that Professor Wumpus has not moved.

IDENTIFIER: 7

DESCRIPTION: To test that once the Student finds the Assignment, they cannot pick it up again.
PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate by following the path: east, east, east, south (You should end at [1,3] 2nd row, 4th column). The game will print “You found the assignment!”.
2) Move west once and move back into the position where you initially found the assignment [2, 4]. The game shall not reproduce the message “You found the assignment!”.

POSTCONDITIONS: After following the first step, the game indicates that the Student has found the assignment. Moving west then back to the position where the assignment initially was does not reproduce the same message indicating the assignment was already found.

IDENTIFIER: 8

DESCRIPTION: The player wins when they hand the assignment to Professor Wumpus, and the game exits cleanly.

PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate by following the path: east, east, east, south (You should end at [1,3] 2nd row, 4th column). The game will print “You found the assignment!”.
2) Follow the path: south, south, south to find Professor Wumpus.

POSTCONDITIONS: The game prints “You turn in your assignment. YOU WIN!” and terminates the game.

IDENTIFIER: 9

DESCRIPTION: To test that if the Student encounters Professor Wumpus but has not found the Assignment, the player will lose.

PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate by following the path: south, south, south, south, east, east, east.

POSTCONDITIONS: The game prints “Prof Wumpus sees you, but you don’t have your assignment. YOU LOSE!” and terminates the game.

IDENTIFIER: 10

DESCRIPTION: To test that the Student hears that Professor Wumpus is pontificating if the player is directly North, South, East or West of Professor Wumpus.

PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate to the point [4, 3] 5th row, 4th column using the path: south, south, south, south, east, east. The game should print “You hear someone pontificating on Computer Science … Professor Wumpus must be nearby!”.
2) Move north once and east once. The game should reproduce the message “You hear someone pontificating on Computer Science … Professor Wumpus must be nearby!”.
3) Move east once and south once. The game will reproduce the same message “You hear someone pontificating on Computer Science … Professor Wumpus must be nearby!”.

POSTCONDITIONS: After the first step the Student is west of Professor Wumpus. After the second step, the Student is north of Professor Wumpus and after the third the Student is east of Professor Wumpus. This is indicated by the message “You hear someone pontificating on Computer Science… Professor Wumpus must be nearby!”

IDENTIFIER: 11

DESCRIPTION: To test that the Student hears the rustling of graded papers when the player is directly North, South, East or West of the TA.
PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

1) Navigate to the point [1,4], 2nd row and 5th column by following the path: east, east, east, east, south.

POSTCONDITIONS: The game prints “You hear the shuffling of graded papers… the TA must be nearby!” because the TA is located at [1,4].

IDENTIFIER: 12

DESCRIPTION: To test that the Student hears the rustling of graded papers when the player is directly North, South, East or West of the TA.
PRECONDITIONS:

- The current working directory in a terminal or command prompt is the path of the **profwumpus.jar** file
- The following command has been entered into the terminal: **java -jar profwumpus.jar 100**

EXECUTION STEPS:

2) Navigate to the point [1,4], 2nd row and 5th column by following the path: east, east, east, east, south.

POSTCONDITIONS: The game prints “You hear the shuffling of graded papers… the TA must be nearby!” because the TA is located at [1,4].

IDENTIFIER: 13

DESCRIPTION: To ensure that the game map is the proper size as specified in requirement 1, as a 6x6 grid.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar** into the command prompt terminal.

2) Input the following path, with each entry followed by hitting Enter: **s, s, s, s, s, s** (s for South).

POSTCONDITIONS: The program generates a map indicating a 6x6 grid graphically. You should hit a wall upon the final “s” input.

IDENTIFIER: 14

DESCRIPTION: This is to test that the game accepts and moves the character appropriately, regardless of the case of valid input.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar** into the command prompt terminal.
2) Input the following path, followed by hitting Enter for each entry: s (south), e (east), n (north), w (west).
3) Input the following path, followed by hitting Enter for each entry: S (south), E (east), N (north), W (west).

POSTCONDITIONS: The player should move in circle by going down 1 room, then right 1 room, then up 1 room, then left 1 room. They should return to their starting room. This should happen once for step 2, and once for step 3.

IDENTIFIER: 15

DESCRIPTION: This is to test that the game rejects invalid movement input and does not change the game state.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar** into the command prompt terminal.
2) Input the following, followed by hitting Enter: foo

POSTCONDITIONS: The player should not move, and the state of the game should not change. The game should print “

IDENTIFIER: 16

DESCRIPTION: This is to test that the game responds appropriately according to user movement input, and the game state changes properly
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar** into the command prompt terminal.
2) Input the following, followed by hitting Enter: s

3) Input the following, followed by hitting Enter: e
4) Input the following, followed by hitting Enter: n
5) Input the following, followed by hitting Enter: w

POSTCONDITIONS: The player should move south one room after step 2, then move right one room after step 3, then move up one room after step 4, then move left one room after step 5.

IDENTIFIER: 17

DESCRIPTION: This is to test that the game rejects the user's attempt to move outside the game boundaries, and that the state of the game does not change.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar** into the command prompt terminal.
2) Input the following, followed by hitting Enter: w
3) Input the following, followed by hitting Enter: e, e, e, e, e
4) Input the following, followed by hitting Enter: n
5) Input the following, followed by hitting Enter: s, s, s, s, s,

POSTCONDITIONS: The player should move south one room after step 2, then move right one room after step 3, then move up one room after step 4, then move left one room after step 5.

IDENTIFIER: 18

DESCRIPTION: This is the test that the program accepts the maximum integer value in Java as the seed argument.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

1) Enter **java -jar profwumpus.jar 2147483647** into the command prompt terminal.

POSTCONDITIONS: The game should accept the seed value, and the game map will be drawn with the Student at [0][0].

IDENTIFIER: 19

DESCRIPTION: This is the test that the program accepts the minimum integer value in Java as the seed argument.
PRECONDITIONS:

- A terminal or command prompt window is open.
- The current working directory is the path of the **profwumpus.jar** file.

EXECUTION STEPS:

2) Enter **java -jar profwumpus.jar -2147483648** into the command prompt terminal.

POSTCONDITIONS: The game should accept the seed value, and the game map will be drawn with the Student at [0][0].

3. TRACEABILITY MATRIX

Requirement Number	Test Case ID
1	13
2	15
3	14
4	16
5	17
6	18, 19
7	0, 1, 2, 3
8	4, 5, 6
9	7, 8, 9
10	10
11	11

4. DEFECTS FOUND

DEFECT #1:

SUMMARY: Game operates as a 5 by 5 matrix, not a 6 by 6 matrix.
DESCRIPTION: When playing the game, the game starts out as a 5 by 5 matrix of rooms. This violates the first requirement stating that the game shall consist of a 6 by 6 matrix of rooms.
REPRODUCTION STEPS:

1) Start game by entering **java -jar profwumpus.jar 100** into an open command prompt or terminal, where **100** represents the seed number.

EXPECTED BEHAVIOR: The game starts by printing a 6 by 6 matrix of rooms.
OBSERVED BEHAVIOR: The game starts by printing a 5 by 5 matrix of rooms.

DEFECT #2:

SUMMARY: Game crashes when hitting the eastern wall from any row.
DESCRIPTION: When the Student is in the top right corner room and moves East, the game will crash and produce an index out of bounds exception. This violates the fifth requirement stating that if a room does not exist, the game will indicate to the player that they cannot move in that direction.
REPRODUCTION STEPS:

1) Start game by entering **java -jar profwumpus.jar 100** into an open command prompt or terminal, where **100** represents the seed number.
2) Enter "E" until you are in the top right corner room (Path should be east, east, east east).
3) Enter "E" once more to move east.

EXPECTED BEHAVIOR: The game shall indicate that the player is unable to go east with "There's a wall there, buddy!".
OBSERVED BEHAVIOR: The game crashes and produces an ArrayIndexOutOfBounds exception.

DEFECT #3:

SUMMARY: Game crashes when entering an invalid seed argument.
DESCRIPTION: When a string, invalid character (i.e. a "s"), or a number greater than the max for a 32-bit signed integer is entered (or an integer less than -2147483648), the game crashes. After the game crashes, it produces a NumberFormatException.
REPRODUCTION STEPS:

1) Start game by entering "java -jar profwumpus.jar howdy" (without quotes) into the command prompt/terminal where the string "howdy" is entered as the seed.

EXPECTED BEHAVIOR: The program ignores the command and runs as if no seed was entered.
OBSERVED BEHAVIOR: The program prints, "Welcome to Professor Wumpus" and throws a NumberFormat exception error.