CS 1632 - DELIVERABLE 1:
Test Plan and Traceability Matrix

Professor Wumpus

Reed Armstrong and James Hahn

Introduction

During this first deliverable, the team came across several concerns and difficulties while designing the test plan and carrying out a run to find defects. We discuss a few of these below.

The Professor Wumpus game allows the TA to hold several responsibilities and actions. For example, if the TA encounters the student in the same room, the student is randomly "teleported" to a different room. Also, the TA moves in a random direction for each iteration of the game, and is "invisible" to the user - they can't be seen on the game's map. The "invisible" property and "random" movement of the TA paired together causes a major problem for the people testing the program. If the TA cannot be seen, how is the randomness supposed to be tested? The TA could move in the same path for every seed, where the path is determined by whoever implemented the game. Even so, this means investigating particular game seeds to try to track and document the behavior of this invisible agent. One has to test extensively until they find the path of the TA over several iterations and then compare it to the path of a TA using a different seed to make sure the seeds do make a difference.

As far as edge cases are concerned, any part of the program that takes input (in our case, the seed and the student's movement) were considered first because they give the user the most control over what is fed into the program. Because of this, for the seed, you can test INT_MAX, INT_MIN, Strings, and other non-integer data types; user input can be considered non-visual edge cases because they are input entered by the user to be handled internally by the application. One defect we found, the student's movement into a wall (in our case, the east wall), represents a more visual sort of edge case. One can see the student along the east border of the matrix of rooms, but the program should prevent the user from moving any further east in the matrix; this was not handled and the program instead threw an ArrayIndexOutOfBoundsException.

Lastly, deciding on what the team tested depended heavily on the general features of the requirements. In order to fully determine the key features of the program, we had to extract the most important aspects to test. For example, to test being able to input lowercase and uppercase letters, obviously those deal with two different character sets, so one test case was written for each. Other tests dealt with winning and losing with the assignment. As one can see, these two tests essentially test opposite cases, both dealing with the same requirement, but they are *key* features of the requirement. This is how most of the test cases in the test plan were chosen; instead of testing every single possible scenario to come across in the program, the most important paths, scenarios, and cases were selected.

Test Cases

IDENTIFIER: room_exists
TEST CASE: The student should be able to move in all 4 directions to rooms that exist.
PRECONDITIONS: The game has started and the program is waiting for the user to enter a command. The top-left corner room of the matrix has coordinates (0, 0) in the format (x, y) and the bottom-right corner room of the matrix has coordinates (5, 5) in the format (x, y).
EXECUTION STEPS: Press the ‘s’ key and then the Enter key.
POSTCONDITIONS: The y coordinate of the new room should be 1 more than the previous room. This can be seen on the matrix: the student is now one room lower than their starting location.

IDENTIFIER: user_input_case_insensitive_lowercase
TEST CASE: The student should be able to move in all 4 directions (North, South, East, West) after any of the corresponding lowercase inputs are given to the program (‘n’, ‘s, ‘e’, or ‘w’) and the Enter key is pressed.
PRECONDITIONS: The game has started and the program is waiting for the user to enter a command. The top-left corner room of the matrix has coordinates (0, 0) in the format (x, y) and the bottom-right corner room of the matrix has coordinates (5, 5) in the format (x, y).
EXECUTION STEPS: Press the ‘e’ key and then the Enter key.
POSTCONDITIONS: The x coordinate of the new room should be 1 more than the previous room, which means the input of ‘e’ was accepted. This will be reflected in the matrix: the student appears one room to the right of their starting location.

IDENTIFIER: user_input_case_insensitive_uppercase
TEST CASE: The student should be able to move in all 4 directions (North, South, East, West) after any of the corresponding uppercase inputs are given to the program (‘N’, ‘S’, ‘E’, or ‘W’) and the Enter key is pressed.
PRECONDITIONS: The game has started and the program is waiting for the user to enter a command. The top-left corner room of the matrix has coordinates (0, 0) in the format (x, y) and the bottom-right corner room of the matrix has coordinates (5, 5) in the format (x, y).
EXECUTION STEPS: Press the ‘E’ key and then the Enter key.
POSTCONDITIONS: The x coordinate of the new room should be 1 more than the previous room, which means the input of ‘E’ was accepted. This will be reflected in the matrix: the student appears one room to the right of their starting location.

IDENTIFIER: room_does_not_exist
TEST CASE: When the student is on the border of the matrix (in a 5x5 matrix, the x or y coordinate is 0 or 4), and user input forces the Student to move further into that border, ensure

that the student does not change rooms and receives a message saying why no move was made.
PRECONDITIONS: The game has started and the program is waiting for the user to enter a command. The student is standing at (0, 0) on the matrix of rooms.
EXECUTION STEPS: Press the 'w' key and then the Enter key.
POSTCONDITIONS: The student does not change rooms, and the program outputs "There's a wall there, buddy!"

IDENTIFIER: random_seed_lower_bound
TEST CASE: Enter the smallest 32-bit signed integer as the command line argument for the game's random seed.
PRECONDITIONS: Start up a command prompt/terminal from a machine and navigate to the directory of the program's .jar file.
EXECUTION STEPS: Enter the command "java -jar profwumpus.jar -2147483648"
POSTCONDITIONS: The program starts the game with the seed -2147483648 as the random number seed.

IDENTIFIER: random_seed_past_lower_bound
TEST CASE: Enter a number 1 smaller than the smallest 32-bit signed integer as the command line argument for the game's random seed. *-Edge Case-*
PRECONDITIONS: Start up a command prompt/terminal from a machine and navigate to the directory of the program's .jar file.
EXECUTION STEPS: Enter the command "java -jar profwumpus.jar -2147483649"
POSTCONDITIONS: The program starts the game, using its own random seed and ignoring the command line argument.

IDENTIFIER: random_seed_upper_bound
TEST CASE: When entering the command to start the program from a command prompt/terminal, enter the largest 32-bit signed integer as a command line argument.
PRECONDITIONS: Start up a command prompt/terminal from a machine and navigate to the directory of the program's .jar file.
EXECUTION STEPS: Enter the command "java -jar profwumpus.jar 2147483647"
POSTCONDITIONS: The program starts the game with the seed 2147483647 as the random number seed.

IDENTIFIER: random_seed_past_upper_bound
TEST CASE: Enter a number 1 greater than the largest 32-bit signed integer as the command line argument for the game's random seed. *-Edge Case-*
PRECONDITIONS: Start up a command prompt/terminal from a machine and navigate to the directory of the program's .jar file.

EXECUTION STEPS: Enter the command “java -jar profwumpus.jar 2147483648”
POSTCONDITIONS: The program starts the game, using its own random seed and ignoring the command line argument.

IDENTIFIER: 6_by_6_matrix
TEST CASE: When the program is running, at every iteration, the user sees a 6x6 matrix of rooms. A room is defined as a pair of brackets separated by 2 spaces. For example: [ ] is 1 room, [ ][ ] is 2 rooms, and [ ][ ] is a 2x2 matrix of rooms, for a total of 4 rooms.

[ ][ ]

PRECONDITIONS: Run the program from a command prompt/terminal using the command “java -jar profwumpus.jar”.
EXECUTION STEPS: Look at the output of the program in the command prompt/terminal.
POSTCONDITIONS: The program outputs a 6x6 matrix of rooms to the screen as part of starting the game.

IDENTIFIER: user_wins_with_assignment
TEST CASE: If the student navigates to the room containing the assignment before encountering professor Wumpus, and the student then navigates to the room of professor wumpus, the user wins the game and the program is immediately stopped.
PRECONDITIONS: Start up the program with the seed -1 and navigate to the room (3, 4) with the path east, east, east, south, south, south, south in a 0-based (x, y) coordinate system. The program outputs “You found the assignment!” indicating the student is currently holding the assignment.
EXECUTION STEPS: Navigate to the room located at (0, 3) with the path west, west, west, north to find professor wumpus.
POSTCONDITIONS: The program outputs “You turn in your assignment. YOU WIN!” and immediately terminates.

IDENTIFIER: user_loses_without_assignment
TEST CASE: If the student navigates to the room containing Professor Wumpus before obtaining the assignment, the user loses the game and the program is immediately terminated.
PRECONDITIONS: Start up the program with the seed -1.
EXECUTION STEPS: Navigate to the room (0, 3) with the path south, south, south.
POSTCONDITIONS: The program outputs “Prof Wumpus sees you, but you don’t have your assignment. YOU LOSE!” and immediately terminates.

IDENTIFIER: matrix_displayed_at_each_iteration
TEST CASE: At every iteration of the program, the program outputs a matrix of rooms to the user, prompting the user for input to move the student.
PRECONDITIONS: Start up the program with any seed.

EXECUTION STEPS: Enter any valid input to advance the program's main game loop by 1 iteration.
POSTCONDITIONS: The program outputs a matrix of rooms of size N by N to the user.

IDENTIFIER: pass_string_as_seed_in_command_prompt
TEST CASE: When starting up the program, a string entered in place of an integer for the seed should cause the program to disregard the command-line argument and proceed normally.
PRECONDITIONS: Start up the program by entering the following command into a command prompt/terminal: "java -jar profwumpus.jar boompow" (without the quotes). *-Edge Case-*
EXECUTION STEPS: Wait for the program to process the command-line arguments and begin the game.
POSTCONDITIONS: The program disregards the "boompow" command-line argument and chooses a random signed integer as the seed and continues the game, displaying a matrix of rooms and waiting for user input.

IDENTIFIER: student_hears_professor_wumpus_from_side
TEST CASE: The student hears that somebody is pontificating on computer science if they are in the room west of professor wumpus.
PRECONDITIONS: Start up the program with the seed 477054061.
EXECUTION STEPS: Navigate to room (1, 2) with the path: east, south, south, in a 0-based (x, y) coordinate system; this is the room west of Professor Wumpus.
POSTCONDITIONS: The program outputs "You hear someone pontificating on Computer Science… Professor Wumpus must be nearby!"

IDENTIFIER: ta_moves_randomly
TEST CASE: For each iteration of the game, the TA moves in a random direction.
PRECONDITIONS: Start up the program with any seed.
EXECUTION STEPS: Navigate to the room located at (3, 2) using the path: east, east, south, east, south. The program outputs "You hear the shuffling of graded papers… the TA must be nearby!", which indicates the TA is to the north, south, east, or west of the student. From that room, try moving to the north, east, south, or west to "catch" the TA. Several restarts of the game may be required to see this to the full effect.
POSTCONDITIONS: The student does not collide with the TA regardless of which direction they move in, proving that the TA has moved from their previous position randomly to a new direction.

IDENTIFIER: accept_direction_to_move
TEST CASE: Ensure that the user can enter one of the four directions to move the student in that direction.
PRECONDITIONS: Start up the program with the seed 1314240236.

EXECUTION STEPS: Enter “s”.
POSTCONDITIONS: The program prints the matrix of rooms, with the student’s position updated to be one lower than the starting position.

IDENTIFIER: reject_spelled_out_directions
TEST CASE: Ensure that if the user enters the full spelling of a cardinal direction to move the student, the program does not accept this and does not move the student.
PRECONDITIONS: Start up the program with the seed 1314240236.
EXECUTION STEPS: Enter “south”. Observe the produced output. Then enter “e”.
POSTCONDITIONS: The program displays “Please enter N, S, E, or W” and does not display the matrix of rooms showing an update to the player’s position after “south” is entered. Then after “e” is entered, the program prints the matrix of rooms showing that the student has moved to the east, and that no movement to the south occurred.

IDENTIFIER: reject_strings_that_are_not_directions
TEST CASE: Ensure that if the user enters a string that cannot be interpreted as a cardinal direction, the program does not accept this as a direction and does not move the student.
PRECONDITIONS: Start up the program with the seed 1314240236.
EXECUTION STEPS: Enter “asdf”. Observe the produced output. Then enter “e”.
POSTCONDITIONS: The program displays “Please enter N, S, E, or W” and does not display the matrix of rooms showing an update to the player’s position after “asdf” is entered. Then after “e” is entered, the program prints the matrix of rooms showing that the student has moved to the east, and that no movement to the south occurred.

IDENTIFIER: student_hears_ta_from_side
TEST CASE: The student hears that somebody is shuffling graded papers if the TA is in a room directly to the north, south, west, or east of the student.
PRECONDITIONS: Start up the program with the seed 477054061.
EXECUTION STEPS: Take this path: east, east, east, south, south, south, south, north, north, south in a 0-based (x, y) coordinate system.
POSTCONDITIONS: The program outputs “You hear the shuffling of graded papers… the TA must be nearby!” because the TA is in the room located at (3, 4).

IDENTIFIER: ta_hits_walls
TEST CASE: Ensure that if the TA were to try to move into a room that does not exist, the TA does not move, and instead hits a wall.
PRECONDITIONS: Start up the game with the seed -272790948.
EXECUTION STEPS: Take this path: east, east, east, east. Observe the output.

POSTCONDITIONS: Three times in a row (on the last three movements), the program displays "You hear a thud, as if the TA hit into a Southern wall…". This indicates that rather than passing through walls, the TA is able to hit them repeatedly and remain in-bounds.

IDENTIFIER: professor_wumpus_does_not_move
TEST CASE: Ensure that Professor Wumpus does not move, and always stays in the same room.
PRECONDITIONS: Start up the game with the seed 516927037.
EXECUTION STEPS: Take this path: south, south, south, south, east, east, east, north. Observe the output.
POSTCONDITIONS: After the final movement eastward, the program displays "You hear someone pontificating on Computer Science… Professor Wumpus must be nearby!". Then after moving north, the program does not display "You hear someone pontificating on Computer Science… Professor Wumpus must be nearby!" again because Professor Wumpus did not move.

IDENTIFIER: student_can_pick_up_assignment
TEST CASE: Ensure that the student can pick up the assignment by moving to the room where it is located.
PRECONDITIONS: The program has started using 2 as the random seed.
EXECUTION STEPS: Move south four times. Observe the output. Then move east once, then west once.
POSTCONDITIONS: After moving all the way south, the program produces the message "You found the assignment!" Then after moving east and back west, the program does not produce the message "You found the assignment!" because the assignment is already collected.

Traceability Matrix

Requirements Reference (with their respective identifiers):
https://github.com/laboon/CS1632_Fall2016/blob/master/deliverables/1/requirements.txt

Requirement Number	**Corresponding Test Cases**
1	6_by_6_matrix matrix_displayed_at_each_iteration
2	accept_direction_to_move reject_spelled_out_directions reject_strings_that_are_not_directions
3	user_input_case_insensitive_lowercase user_input_case_insensitive_uppercase
4	room_exists
5	room_does_not_exist
6	random_seed_lower_bound random_seed_upper_bound
7	random_seed_past_lower_bound - *edge case* random_seed_past_upper_bound - *edge case* pass_string_as_seed_in_command_prompt - *edge case*
8	ta_moves_randomly ta_hits_walls professor_wumpus_does_not_move
9	user_wins_with_assignment user_loses_without_assignment student_can_pick_up_assignment
10	student_hears_professor_wumpus_from_side
11	student_hears_ta_from_side

Defects

SUMMARY: Index out of bounds when moving east into a wall
DESCRIPTION: When in the bottom right corner of the map, if you input ‘e’ to move east, it gives you an index out of bounds exception.
REPRODUCTION STEPS: Start up the program with 100 as the seed. Then, move the student to the bottom right corner with the path of (east, east, east, east, south, south, south, south), and finally enter ‘e’ to move east.
EXPECTED BEHAVIOR: The program outputs “There’s a wall there, buddy!”, redraws the map with the player still located in the bottom right corner, and prompts for another movement.
OBSERVED BEHAVIOR: An ArrayIndexOutOfBoundsException is thrown and the program crashes.

SUMMARY: Invalid seed causes the program to crash
DESCRIPTION: When a string is entered as a command-line argument for the pre-determined seed, the program crashes.
REPRODUCTION STEPS: Navigate to the directory of your profwumpus.jar file using a command prompt/terminal and enter the following command into the command prompt/terminal: “java -jar profwumpus.jar boompow” (without the quotes). Note here that the string “boompow” is being entered as a command-line argument for the seed of the program.
EXPECTED BEHAVIOR: The program ignores the command-line argument for the seed and continues as if no seed was entered in the first place.
OBSERVED BEHAVIOR: The program prints “Welcome to Professor Wumpus” and immediately throws a NumberFormatException on the next line and crashes.

SUMMARY: The game is a 5x5 matrix instead of 6x6
DESCRIPTION: Once the game is started, there is a matrix that is 5 rooms horizontally by 5 rooms vertically instead of 6 rooms horizontally by 6 rooms vertically.
REPRODUCTION STEPS: Start the game by entering in the appropriate command with any seed. Then, look at the number of rooms horizontally as well as vertically. One room is shaped as 2 brackets with 2 spaces between them. For example, [ ] is a room and [ ][ ][ ] is 3 rooms horizontally.
EXPECTED BEHAVIOR: The program prints out a 6x6 matrix of rooms to the user so there are 36 rooms to traverse between.
OBSERVED BEHAVIOR: The program prints out a 5x5 matrix of rooms to the user, thus allowing only 25 rooms to explore.