Mohit Jain

CS 1632

Deliverable #1

**Requirement #1**

TEST CASE IDENTFIER: Matrix-Dimension

DESCRIPTION: This test verifies that the matrix is a 6 by 6 matrix as specified.

PRECONDITIONS: The player just started the game.

EXECUTION STEPS: The program must be run using java –jar profwumpus.jar.

POSTCONDITIONS: The program displays a 6 by 6 matrix, and the current location starts from the first entry of this matrix.

**Requirement #2**

TEST CASE IDENTFIER: Check-Input-NSEW

DESCRIPTION: This test verifies that when the user types in an input other than N, S, E, or W, then the program outputs: “Please Enter N, S, E, or W”.

PRECONDITIONS: The player just started the game.

EXECUTION STEPS: The user enters the input “NSEW”.

POSTCONDITIONS: The program displays the following text: “Please enter N, S, E, or W”.

TEST CASE IDENTFIER: Check-Input-0

DESCRIPTION: This test verifies that when the user types in an input other than N, S, E, or W, then the program outputs: “Please Enter N, S, E, or W”.

PRECONDITIONS: The player just started the game.

EXECUTION STEPS: The user enters the input “0”.

POSTCONDITIONS: The program displays the following text: “Please enter N, S, E, or W”.

TEST CASE IDENTFIER: Check-Input-NN

DESCRIPTION: This test verifies that when the user types in an input other than N, S, E, or W, then the program outputs: “Please Enter N, S, E, or W”.

PRECONDITIONS: The player just started the game.

EXECUTION STEPS: The user enters the input “NN”.

POSTCONDITIONS: The program displays the following text: “Please enter N, S, E, or W”.

**Requirement #3**

TEST CASE IDENTFIER: Check-Case-Input-N

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “N”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-W

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “W”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-E

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “E”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the east. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-S

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet, and the text is displayed that follows “You smell the ink of a printed assignment”.

EXECUTION STEPS: The user enters the input “S”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the south. It also displays the text “You found the assignment”. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-n

DESCRIPTION: This test verifies that when the user types in an input that includes n, s, e, or w, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “n”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-w

DESCRIPTION: This test verifies that when the user types in an input that includes n, s, e, or w, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “w”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-e

DESCRIPTION: This test verifies that when the user types in an input that includes n, s, e, or w, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “e”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the east. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Case-Input-s

DESCRIPTION: This test verifies that when the user types in an input that includes n, s, e, or w, then the program will work as specified. This test is mainly to check for case insensitivity.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet, and the text is displayed that follows “You smell the ink of a printed assignment”.

EXECUTION STEPS: The user enters the input “s”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the south. Our current location is properly displayed.

**Requirement #4**

TEST CASE IDENTFIER: Check-Iteration-Input-E

DESCRIPTION: This test verifies that when the user types in an input that allow them to move one direction into the east, then the matrix will show our current location moved one direction to the right.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “E”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the east. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Iteration-Input-S

DESCRIPTION: This test verifies that when the user types in an input that allow them to move one direction into the south, then the matrix will show our current location moved one direction down the matrix.

PRECONDITIONS: At this point, in the program we are one step to the east direction from the starting position, i.e. in the second entry (a12) of the first row in the matrix.

EXECUTION STEPS: The user enters the input “S”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the south. Our current location is properly displayed at a22.

TEST CASE IDENTFIER: Check-Iteration-Input-W

DESCRIPTION: This test verifies that when the user types in an input that allow them to move one direction into the west, then the matrix will show our current location moved one direction to the left of the matrix.

PRECONDITIONS: At this point, in the program we are at the matrix entry a22.

EXECUTION STEPS: The user enters the input “W”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the west. Our current location is properly displayed at a21.

TEST CASE IDENTFIER: Check-Iteration-Input-N

DESCRIPTION: This test verifies that when the user types in an input that allow them to move one direction into the north, then the matrix will show our current location moved one direction up the matrix.

PRECONDITIONS: At this point, in the program we are at the matrix entry a21.

EXECUTION STEPS: The user enters the input “N”.

POSTCONDITIONS: The program displays the matrix including our current location by moving our current location one space toward the north. Our current location is properly displayed at a11.

**Requirement #5**

TEST CASE IDENTFIER: Check-Wall-Input-N

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W and the player is near an edge in which they cannot move up, down, left, right because a room is not specified in that direction, then the program will not allow them to move in such a direction.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “N”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Wall-Input-W

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W and the player is near an edge in which they cannot move up, down, left, right because a room is not specified in that direction, then the program will not allow them to move in such a direction.

PRECONDITIONS: At this point, the program is at the very start, i.e. the current location is at the very first entry and has not gone anywhere yet.

EXECUTION STEPS: The user enters the input “W”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Wall-Input-E

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W and the player is near an edge in which they cannot move up, down, left, right because a room is not specified in that direction, then the program will not allow them to move in such a direction.

PRECONDITIONS: At this point, the program is at the very eastern end of the final column in the matrix.

EXECUTION STEPS: The user enters the input “E”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

TEST CASE IDENTFIER: Check-Wall-Input-S

DESCRIPTION: This test verifies that when the user types in an input that includes N, S, E, or W and the player is near an edge in which they cannot move up, down, left, right because a room is not specified in that direction, then the program will not allow them to move in such a direction.

PRECONDITIONS: At this point, the program is at the any point in the final row of the matrix.

EXECUTION STEPS: The user enters the input “S”.

POSTCONDITIONS: The program displays the following text “There’s a wall there buddy” and then displays the matrix including our current location. Our current location is properly displayed.

**Requirement #6**

TEST CASE IDENTFIER: Postive-32-bit-Seed-Argument

DESCRIPTION: This test verifies that when the user types in a positive integer that can be converted into a 32-bit signed number, the program will accept that argument.

PRECONDITIONS: The player has not started the game yet.

EXECUTION STEPS: The user enters a positive integer as an input that can be converted into a 32-bit signed number.

POSTCONDITIONS: The program runs as specified without showing error and accepts the argument.

TEST CASE IDENTFIER: Negative-32-bit-Seed-Argument

DESCRIPTION: This test verifies that when the user types in a negative integer that can be converted into a 32-bit signed number, the program will accept that argument.

PRECONDITIONS: The player has not started the game yet.

EXECUTION STEPS: The user enters a negative integer as an input that can be converted into a 32-bit signed number.

POSTCONDITIONS: The program runs as specified without showing error and accepts the argument.

**Requirement #7**

TEST CASE IDENTFIER: Greater-Than-32-bit-Seed-Argument

DESCRIPTION: This test verifies that when the user types in any integer positive or negative that cannot be converted into a 32-bit signed number, the program will invalidate that input and run by assuming that no input was passed in.

PRECONDITIONS: The player has not started the game yet.

EXECUTION STEPS: The user enters a positive or negative integer as an input that is greater than 32 bits.

POSTCONDITIONS: The program runs and ignores any argument(s) entered.

TEST CASE IDENTFIER: Not-An-Integer

DESCRIPTION: This test verifies that when the user types in an input that is not an integer (it is a letter or a word), the program will invalidate that input and run by assuming that no input was passed in.

PRECONDITIONS: The player has not started the game yet.

EXECUTION STEPS: The user enters a word or a letter of the alphabet as an input.

POSTCONDITIONS: The program runs and ignores any argument(s) entered.

**Requirement #8**

TEST CASE IDENTFIER: TA-Walk-Around

DESCRIPTION: This test verifies that when the user initially finds the TA in an arbitrary location in the matrix that when the user comes back around to the same location in which he had previously found the TA, he will not have found the TA this time. The test overall verifies that the TA is constantly moving around and changing location instead of just staying in one spot throughout the game.

PRECONDITIONS: The user met the TA at a certain spot in the matrix.

EXECUTION STEPS: Once the precondition has been fulfilled, and the user has been placed in an arbitrary location, the user must go back to the location he previously met the TA to check if the TA still resides in that location

POSTCONDITIONS: The user will not meet the TA once he has gone to that exact location he was previously in when he met the TA.

TEST CASE IDENTFIER: TA-Bumps-Into-Wall

DESCRIPTION: This test verifies that when the user is walking around the matrix that at some point, the user is informed that the TA has bumped into wall.

PRECONDITIONS: The player needs to be walking around the matrix in arbitrary directions.

EXECUTION STEPS: The user enters north, south, east or west to indicate the direction for which he needs to move, and continues doing so till he gets a message that the TA bumped into a wall.

POSTCONDITIONS: The program lets you know the TA hit into a wall.

TEST CASE IDENTFIER: Stationary-Wumpus

DESCRIPTION: This test verifies that Professor Wumpus is stationary, and does not change his location.

PRECONDITIONS: The player needs to first find a notification that tells the player that Professor Wumpus is nearby.

EXECUTION STEPS: After the player finds a notification of Professor Wumpus’ proximity, the player must go back from where he came from and revisit the same location for which he received this notification

POSTCONDITIONS: The program lets you know for the second time that Professor Wumpus is nearby.

**Requirement #9**

TEST CASE IDENTFIER: Assignment-Submitted

DESCRIPTION: This test verifies that when the user has first found the assignment and then found the professor that the program ends and displays that the player has won the game.

PRECONDITIONS: The player has started the game and has found the assignment.

EXECUTION STEPS: The user finds Professor Wumpus and goes to his location.

POSTCONDITIONS: The program ends giving you a notification that you turned in your assignment and goes on to display that you have won the game.

TEST CASE IDENTFIER: My-Dog-Ate-My-Assignment

DESCRIPTION: This test verifies that when the user finds Professor Wumpus without first finding the assignment that the program ends and displays that the player has lost the game.

PRECONDITIONS: The player has started the game and has not found the assignment.

EXECUTION STEPS: The user finds Professor Wumpus and goes to his location.

POSTCONDITIONS: The program ends giving you a notification that you have found Professor Wumpus but do not have the assignment and goes on to display that you have lost the game.

**Requirement #10**