Group 8

Aryan Patel - 301226774

Dixit Hihoriya - 301201312

Test Case Design

It is suggested to work in teams of two (you could use MSTeams to communicate with your team member). You and a partner are developing a program that lets a user play tic-tac-toe against the computer. The goal is to identify test cases before the code is written (using the V-Model). You want to maximize code coverage by testing as many paths through the logic as possible at the same time as controlling time and expense by designing a minimal set of test cases to give you confidence that the code works.

**Marking**: This assignment is marked out of 10.

**Due**: One member of each team must upload an MS Word document giving your solution to the corresponding Assignments folder on eCentennial.

# Rules of the game

|  |  |
| --- | --- |
| |  | | --- | | * The user can choose to play X or O. * X always goes first. * Then X and O take turns placing a marker on the board. * The goal is to win by placing three markers (X-X-X) or (O-O-O) across one row, down one column, or along a diagonal. |   The game often ends in a tie when the nine positions are filled but neither X nor O completed a line. |

# Scope of this exercise

You are engaged in an Agile development process. In this increment, you are focusing on the logic of deciding what move to make. In other words, test the artificial intelligence of a computer playing a game.

Previous increments prepared the board and prototype user interface. You can assume that tests for valid game moves were completed in an earlier increment. For example, assume user or computer don’t try to play into an occupied square or play an X when it’s O’s turn. Similarly, don’t worry about how the board is represented internally or displayed to the user. Just trust that some 3x3 display is mapped onto an appropriate data structure in the code.

# The code you are testing

The logic for computer’s move is to try the following in order:

a. If you have two pieces in a line and can complete a line by placing the third, put your X or O in the winning cell.

b. If the user has two markers in a line and can complete the line on his/her next move, put your marker in the cell that blocks the user from winning.

c. If the middle of the board is free, put a marker there.

d. If at least one corner is free, put a marker in a corner. Choose the corner randomly.

e. Select randomly from any of the unoccupied cells.

**Notes:** The five rules above simplify the logic, and don’t always pick the best move. Therefore, it is possible for the user to beat the computer. The user does not have to follow the same rules and make very silly, but legal, moves.

Note: To analyse these properly you should rely on tried and true methods of information processing. For example, you can come up with a shortcut way of referring to conditions a.-e. above by naming them respectively:

a.-WIN move, b.-BLOCK move, c.-CENTER move, d.-CORNER move and e. SIDE move (or RANDOM move)

# The test objective

The test objective is to verify that the program is applying the rules above in all situations. The goal of this exercise is to design functional tests. Your task is to analyse, design and document enough test cases to be confident of the verification testing. Testing every possibly configuration of pieces is virtually impossible, even for a game as simple as tic-tac-toe.

Your company will outsource your test cases to test specialists who will implement and execute the tests. Testers will compare actual results with expected results and report all deviations as possible defects.

# Instructions

Work with one partner. Healthy discussion and an exchange of different views is good. Finding a consensus approach is part of the exercise. Remember, many different solutions may be good.

Create your solution as an MS Word document.

**First**, decide how many test cases you need and list them. Use a numbered list so that the numbering gives each test case a unique identifier and add a headline-style[[1]](#footnote-1) title (see template below).

|  |  |
| --- | --- |
| Test Case Identifier (ID) | Test Case Title |
| TC1 | Computer, with X, plays B2 (center) |
| TC2 | Computer, with X, plays B1 (side) |
| TC3 | Computer, with X, plays B3 (side) |
| TC4 | Computer, with X, plays C1 (corner) |
| TC5 | Computer, with X, plays C2 (side) |
| TC6 | Computer, with X, plays C3 (corner) |
| TC7 | Computer, with X, plays A1 (corner) |
| TC8 | Computer, with X, plays A2 (side) |
| TC9 | Computer, with X, plays A3 (corner) |

**Note:** To complete the following second task below of this Lab Assignment, whenever you need to Specify a precondition or a post condition, you can do it via using the template below. Make sure you identify via the subscripts the order of movements. For example, in the template below X of course has started 1st and his first move was in B2 (notice the Excel notation is used in this depiction). The PC has used C3 as its first movement; then the player has put his movement in C1 where you see X2, the second movement of player X indicated via X subscript 2 or X2. Second movement of PC is 02 placed in cell A3 and then the player places in B3 its third movement or X3. This can serve as a precondition in the specifications below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **A** | **B** | **C** |
| 1 |  |  | X2 |
| 2 |  | X1 |  |
| 3 | O2 | X3 | O1 |

**Second**, write a detailed specification for at least two test cases. Use the template given:

* **Test case identifier** – from your list of test cases
* **Test case title** -- from you list of test cases
* **Condition** – in this case it is sufficient to specify which of the rules listed above apply. Do not assume there is a one-to-one relationship between test cases and rules. Some rules may justify several different tests and other may not need any.
* **Precondition** – state before computer makes a move. The precondition must be a valid state that could be reached in a game when the program works correctly.
* **Action** – trigger that causes the program to do something. For example, the user asking the computer to make the first move is an action.
* During play, the user’s move sets the precondition and the fact the user completes a turn is the trigger for the computer to take a turn.
* **Postcondition** – expected state after the computer has moved If more than one correct outcome may be possible, show them all.

Do not forget that the computer may be playing as X or as O, and you are testing the movements of computer (not the player – the player cam move as they please), so they have to obey conditions a.-e.

See the template below for an example. While the following example can be in your list in Task One, please do not use this example in Task Two.

First Test Case:

|  |  |
| --- | --- |
| Test Case Identifier | TC1 |
| Test Case Title | Computer, with X, plays B2 (center) |
| Condition | Tests rule c. (CENTER move) |
| Precondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 |  |  |  | | 2 |  |  |  | | 3 |  |  |  | |
| Action | Computer plays first, so given that condition a. does not apply, and so b., it has to apply condition c. (CENTER move), so it plays B2 |
| Postcondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 |  |  |  | | 2 |  | X1 |  | | 3 |  |  |  | |

Second Test Case:

|  |  |
| --- | --- |
| Test Case Identifier | TC9 |
| Test Case Title | Computer, with X, plays A2 |
| Condition | Tests rule B |
| Precondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O1 |  | X2 | | 2 |  | X1 |  | | 3 | O2 |  |  | |
| Action | The user tries to block a win of the computer by placing a marker in A3. The computer uses rule B and place a marker in A2 which blocks a win for the user. |
| Postcondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O1 |  | X2 | | 2 | X3 | X1 |  | | 3 | O2 |  |  | |

Third Test Case:

|  |  |
| --- | --- |
| Test Case Identifier | TC7 |
| Test Case Title | Computer, with X, plays C3 |
| Condition | Tests rule D |
| Precondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O2 |  | X2 | | 2 | X3 | X1 | O3 | | 3 | O1 |  |  | |
| Action | The user blocks a win for computer by placing a marker in C2. The computer uses rule D and places a marker in the available corner C3. |
| Postcondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O2 |  | X2 | | 2 | X3 | X1 | O3 | | 3 | O1 |  | X4 | |

Fourth Test Case:

|  |  |
| --- | --- |
| Test Case Identifier | TC3 |
| Test Case Title | Computer, with X, plays B3 |
| Condition | Tests rule E |
| Precondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O2 | O4 | X2 | | 2 | X3 | X1 | O3 | | 3 | O1 |  | X4 | |
| Action | The user places a marker at B1. The computer is left with one spot so it uses rule E and puts marker at B3. |
| Postcondition | |  |  |  |  | | --- | --- | --- | --- | |  | **A** | **B** | **C** | | 1 | O2 | O4 | X2 | | 2 | X3 | X1 | O3 | | 3 | O1 | X5 | X4 | |

1. A “headline-style title” means “A title that is short but contains the important information” as in headlines of news papers. [↑](#footnote-ref-1)