Test subject: Input taking/interacting with the board

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test | Expected | Actual | Date | Notes |
| Expected cases | Inputting coordinates for a particular cell.  [1, a] | The player can interact with cell [1, a] (in this case flag the cell) |  | 15/05/24 | Coordinate for the y-axis is a string but it has to be an integer to locate a particular cell e.g. board[3][3] |
| Inputting coordinates for a particular cell.  [1, a] | *The player can interact with cell [1, a] (in this case flag the cell)* |  | 15/05/24 | Change the Y coordinate from string to char because char can refer to a number. |
| Inputting different options  [Dig]  [6, e] | After selecting the cell, that cell should be revealed. |  | 15/05/24 |  |
| Inputting different options  [flag]  [9, f] |  |  | 15/05/24 |  |
| Inputting different options  [undo]  [9, f] | The cell that is [F]  Should go back to “o”. | ➡️ | 15/05/24 |  |
|  |  |  |  |  |
| Invalid cases | Inputting coordinates for a particular cell. But using invalid input  [-23, a3!] | Loop through until get a valid input. |  | 11/06/24 |  |
| Inputting an Invalid option  [ fl4, fl%, di, di#, un!, und@ ] | Loop through until get a valid input. |  | 11/06/24 |  |
| null | It should also loop through until it is valid. |  | 11/06/24 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Boundary cases | Inputting  Coordinate at the edge of the table.  [Flag]  [1, a], [10, a]  [1, j], [10, j] | Should be able to work fine. |  | 11/06/24 |  |
| Coordinate of y-axis with capital letters.    [flag]  [5, G] | Should be able to work fine. |  | 11/06/24 |  |
| Inputting different options with random capitalized letters.  [fLAg] | Should be able to work fine. |  | 11/06/24 |  |
| Undo a cell that have no flag  [undo]  [5, h] | The program should tell the player that they can’t undo this cell. |  | 11/06/24 |  |
| Flag the cell that has already been open  [flag]  [1, g] | The program should tell the player they can’t place a flag here. |  | 11/06/24 |  |

Test subject: Revealed a number

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test | Expected | Actual | Date | Notes |
| Expected cases | When opening an empty cell, it should expand out. |  | \*\*stack overflow error. | *2/06/24* | The recursive function that checks the cell around one cell checks each other back and forth causing an infinite loop. |
| When opening an empty cell, it should expand out. |  |  | 2/06/24 | If the cell already checks and is empty change that cell to “0” Then if the cell is “0” return because it has already been checked. |
| When opening a cell that contains a number  [flag]  [7, g] | It should reveal the number. |  | 2/06/24 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Invalid cases |  |  | Invalid cases such as out-of-range coordinates or wrong input for the y-axis (not a-j) when digging are already tested in the input taking part. |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Boundary cases | When digging a cell that has already been flagged  [dig]  [8, a] | Should not reveal the cell. |  | 7/06/24 | It works fine but it should also tell the player that there is a flag there so the player knows. |
| When digging a cell that has already been flagged  [dig]  [8, a] | The program should also tell the player that they have to remove the flag first. |  | 7/06/24 |  |
| When digging a cell that has already been flagged and that cell contain a bomb  [dig]  [9,a] | The player should not lose because there is a flag and they dig. Even though the bomb is there but there is also a flag so the program should tell the player first. |  | 7/06/24 | It should work fine but there is some error in the code. |
| When digging a cell that has already been flagged and that cell contain a bomb  [dig]  [8,d] | The player should not lose because there is a flag and they dig. Even though the bomb is there but there is also a flag so the program should tell the player first. |  | 7/06/24 | The error is because I check if there is a bomb in that cell before if there is a flag so if there is a bomb the player will lose straightway.    I fix it by changing the odder of if/if else statement so the program checks if there is a flag first then check if there a bomb.    It works fine now. |

Test subject: End of game/win/lose

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Test | Expected | Actual | Date | Notes |
| Expected cases | When all the flags are used and the flag is at the correct location. | Tell the player that they won. |  | 10/06/24 |  |
| When all the flags are used and not all the flag is in the correct location | Tell the player that they lost and say how many bombs they missed. |  | 10/06/24 | The flag is at the wrong place but it still says the player wins. There is something wrong |
| When all the flags are used and not all the flag is in the correct location | Tell the player that they lost and say how many bombs they missed. |  | 11/06/24 | Before fixing the error    After fixing the error    The error is because the if statement for checking the flag will run if display[x][y] = “F”. But I use “f” instead the program did not run the check at all. |
| If the player digs a bomb  [dig]  [4, f] | Tell the player that they dug a bomb and they lose. |  | 10/06/24 |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Invalid cases | *e.g. -1* | *“You must enter a number >= 0”* |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Boundary cases | *1/1000000000000000* | *0* |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |