Test Case ID	How I fixed the issue	Evidence (Key Words)
Example	*"At first, when I clicked the Add button, nothing happened. I realised I forgot to connect the command= in the Button to the function. I fixed this by adding command=add_task."*	***command = function()***
4	At first I noticed that some of the buttons were missing and therefore realised that the buttons were overlapping one another having the same coordiante location. I noticed that I was adding two buttons in the same instance of calling the function to make the button widget. Therefore I fixed this by changing when the function was called.	**def button_grid_2(parent_frame):** **button_count = 1** **game_row = 0** **game_column = 0** **game_list = []** **while button_count != 10:** **if button_count <= 3:** **game_row = 0** **elif button_count <= 6:** **game_row = 1** **else:** **game_row = 2** **if button_count == 4:** **game_column = 0** **elif button_count == 5:** **game_column = 1** **elif button_count % 3 == 0:** **game_column = 2** **elif button_count % 2 == 0:** **game_column = 1** **else:** **game_column = 0** **button = ttk.Button(parent_frame, text=f"#{button_count}, R{game_row}, C{game_column}", command=lambda:button_func(button_count))** **button.grid(row=game_row, column=game_column)** **game_list.append(f"#{button_count}, R{game_row}, C{game_column}")** **button_count += 1** **print(game_list)** **return button**
5	At first when I initiated the tkinter window I noticed that the buttons did not instantiate with numbers bu twith the widget id themselvs. I noticed that I did not refer the id arguement to the object in the class. Therefore I added the id variable arguement for each object created via the class.	**self.button = ttk.Button(game_frame, text=self.id, command=self.button_func)**
6	At first when printing the Id into the terminal I noticed that the order of the numbers in the grid did not go numerically. I noticed that I there was an exception of 4 and 5 according to the if statement rules I set for ordering in a 3x3 grid. I fixed this by making 4 and 5 switch manually to fix this grid system.	**if button_count == 4:** **game_column = 0** **elif button_count == 5:** **game_column = 1** **elif button_count % 3 == 0:** **game_column = 2** **elif button_count % 2 == 0:** **game_column = 1** **else:** **game_column = 0**
10	At first when I met the requirment for the 'o's player to win it had not printed a message to confirm this output. I had noticed that I did not change the requirement in terms of 'o' but left it in terms of 'x'. I fixed this by changing the if statement appropriately	**if self.button.cget("text") == "x":** **if self.button.cget("text") == "o":**

11	At first when I was trying to modify the label in terms of the score variables I had made, I ended up with PY_VAR7. This was because I did not refer to the properties of the widget but the widget itself. I fixed this by adding .get() to gain the properties of the widget and therefore update the score.	**cross_score.get()**
12	At first when I changed the tkinter variable describing the score, it did not change when I had met the win condition. I noticed that I did not add the call method for the win condition to make this effect happen. I fixed this by adding the necessary call for the method	**score_add()**
13	At first when I pressed the restart button I noticed that the programme started to lag and crashed. I put a print command to see how many buttons were being pressed and noticed that there were buttons being created for every button on the current window. I fixed this by correctly indenting the command to restart the game and create new buttons.	**def game_start():** **#game_frame.destroy()** **for button in game_frame.winfo_children():** **button.destroy()** **button_grid(game_frame)** **def game_start():** **for button in game_frame.winfo_children():** **button.destroy()** **button_grid(game_frame)**
14	Althought I had fixed the issue where the programme was lagging, there was another issue that when the player only did 1 or so moves after 1 game was completed they had won. I noticed that i did not reset the index of the list of inputs the player had made to compare with the win conditions to allow the player to win. I fixed this by resetting this list through a function.	**def game_set():** **global crosses** **global record_list** **global x_list** **global o_list** **global disable_count** **disable_count = 0** **crosses = True** **record_list = {}** **x_list = []** **o_list = []**
17	When I had tested occupying all buttons to end in a tie the game did not restart. I noticed that I had not correctly recorded if the buttons were being occupied in my original code and therefore made a new variable that just tracks how many buttons have the disabled state until it become 9.	**if disable_count == 9:** **score_add("tie")** **game_start()**