Assignment 1 Testing Documentation

Name: Dylan Brosseau (0724989)
Luka Piplica (0748533)
Opeyemi Okunmuyide (0736535)

Test 1	
Description	Using a 5 by 5 grid with 5 black boxes

Main()	Puzzle puz = new Puzzle(5);
	puz.Initialize(5);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

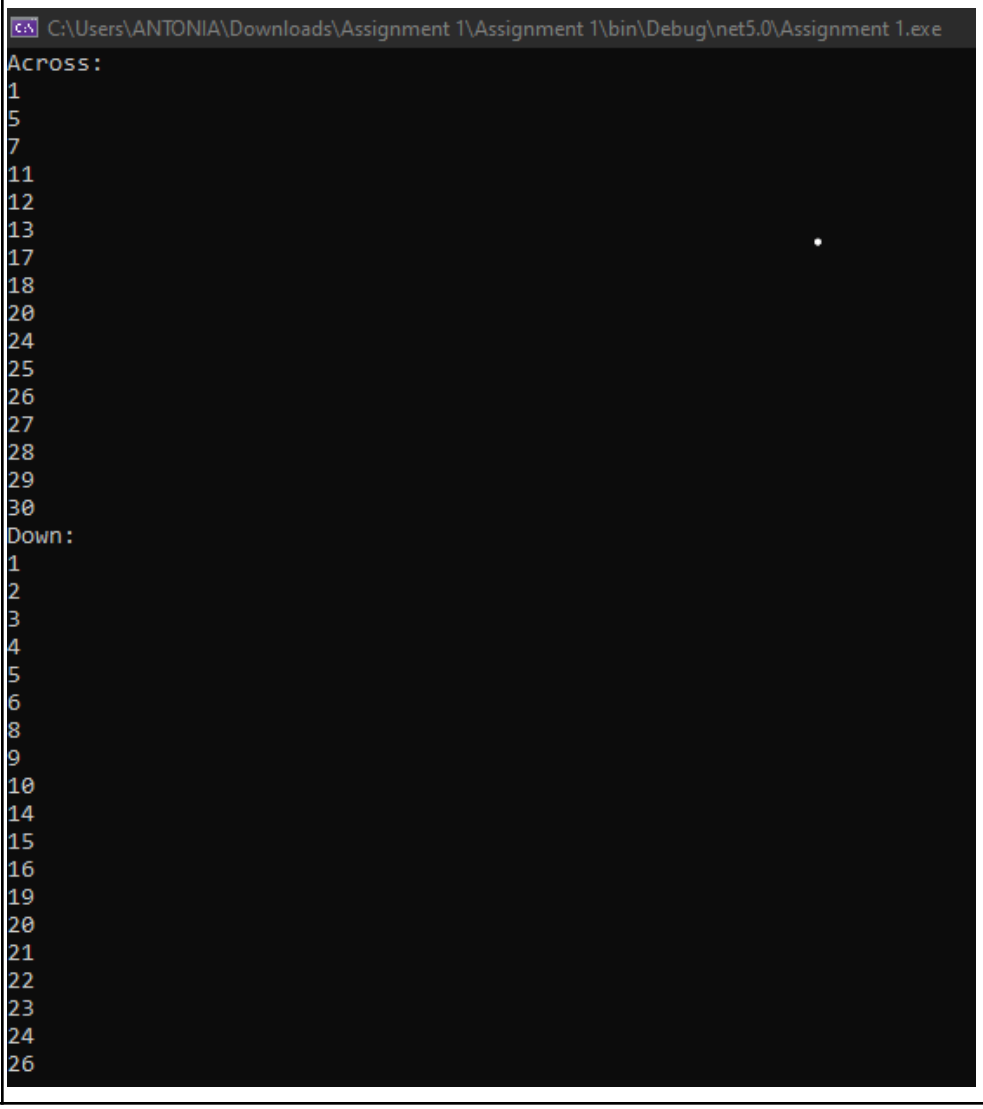
Expected Output	A 5 by 5 grid with 5 randomly placed black boxes and correct lists of clues

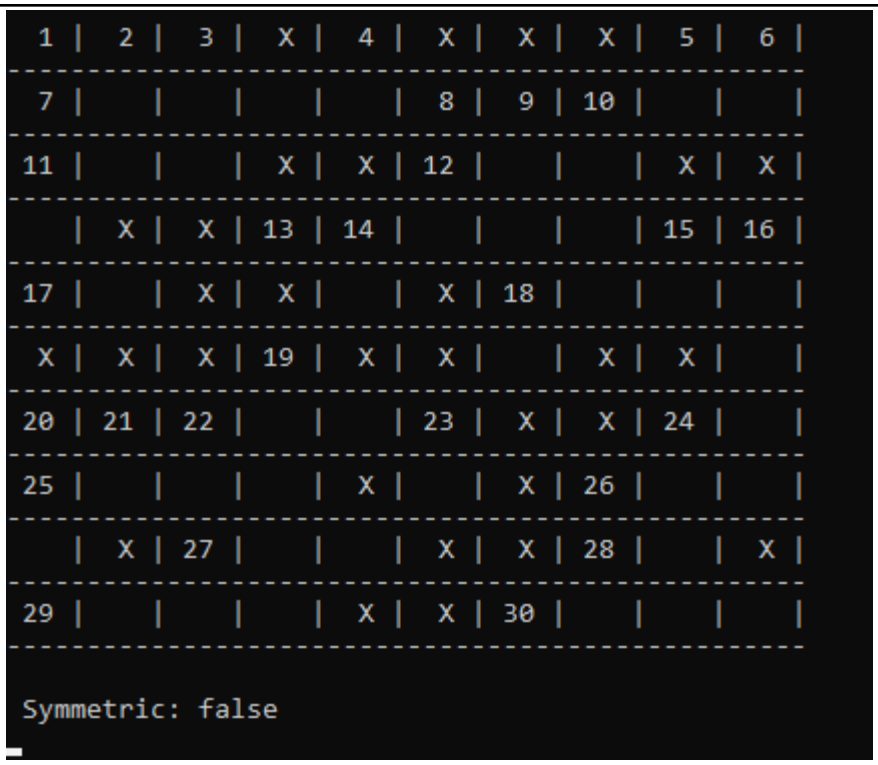
Actual Output	

Test 2	
Description	Using 10 by 10 grid and having a greater number of black boxes

Main()	Puzzle puz = new Puzzle(10);
	puz.Initialize(30);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();

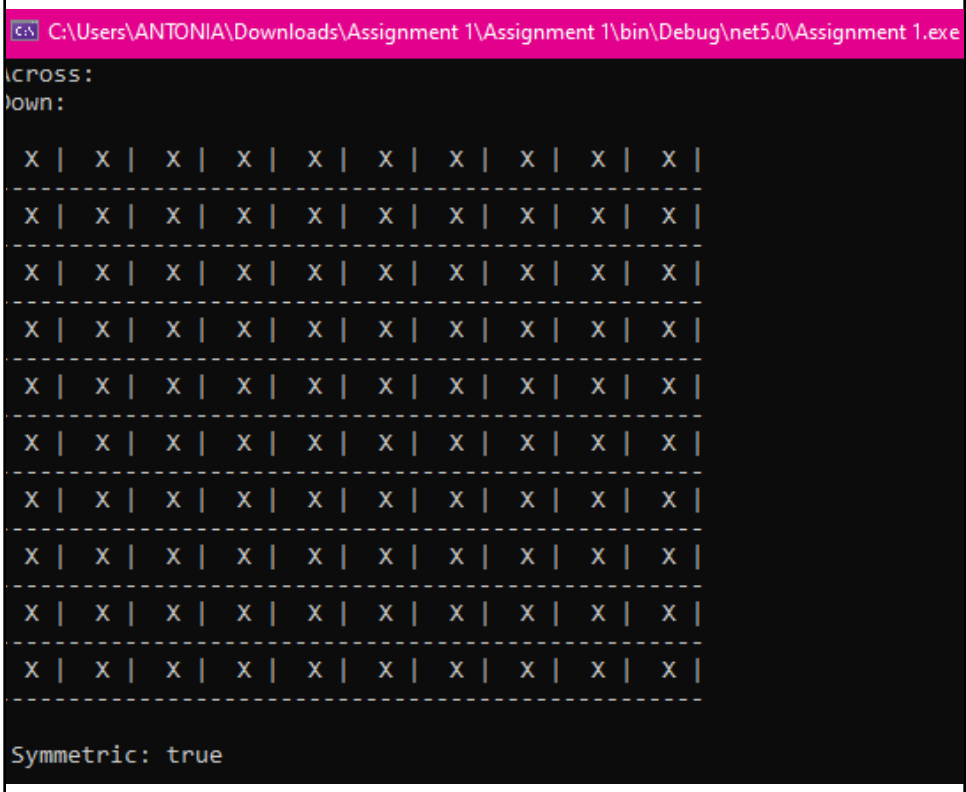
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

Expected Output	A 10 by 10 grid with 30 randomly placed boxes and correct clue lists
Actual Output	



Test 3	
Description	Using a 10 by 10 grid filled with black boxes

Main()	Puzzle puz = new Puzzle(10);
	puz.Initialize(100);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");

	else Console.WriteLine("\n Symmetric: false");
Expected Output	A 10 by 10 grid full of black boxes with no clues
Actual Output	

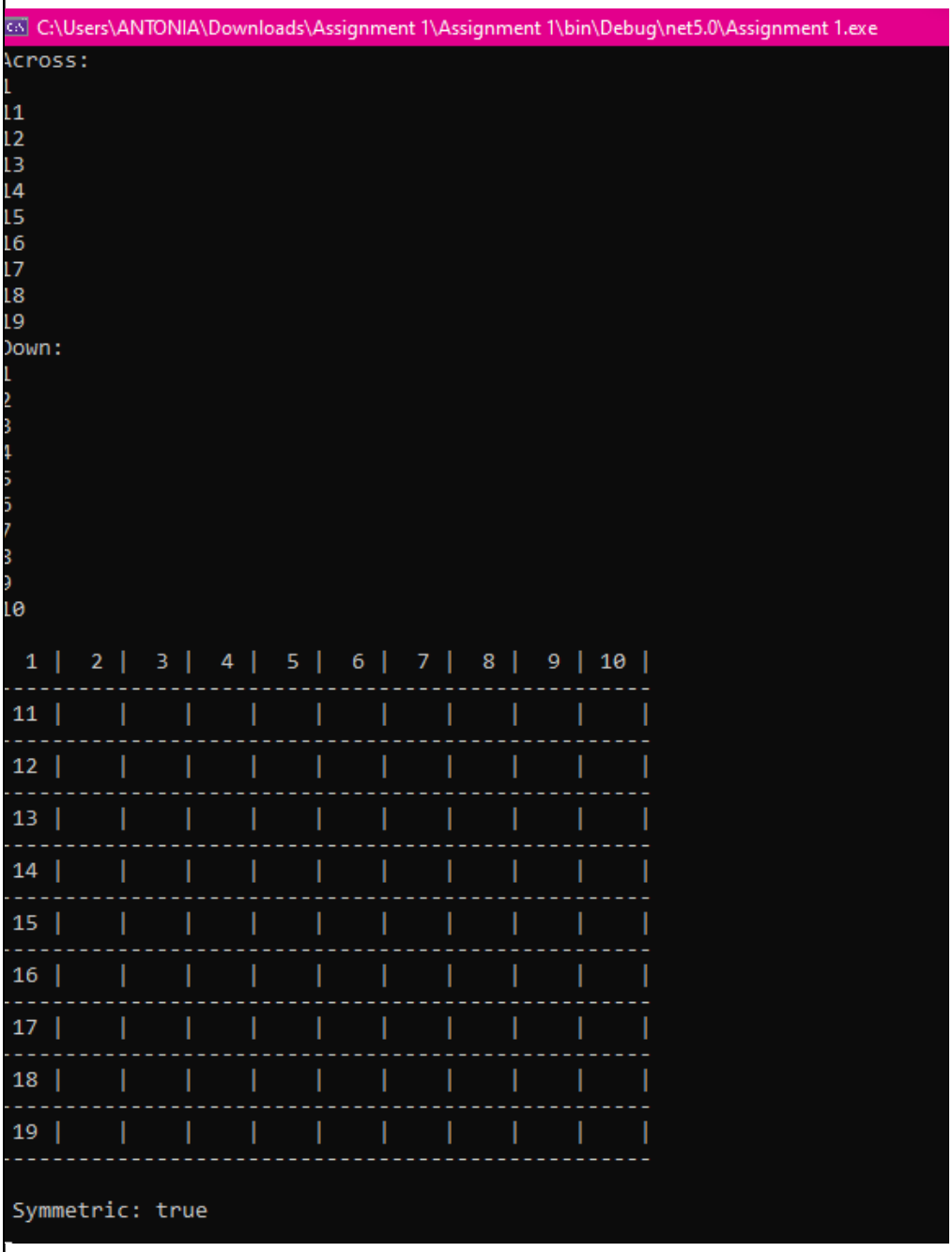
Test 4	
Description	Using 10 by 10 grid with no black boxes

Main()	Puzzle puz = new Puzzle(10);
	puz.Initialize(0);
	puz.Number();

	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

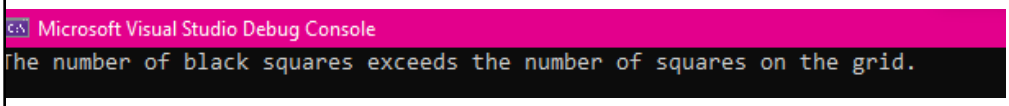
Expected Output	A 10 by 10 grid with no black boxes and 19 clues, with clue 1 going down and across

Actual Output



Test 5	
Description	Using a 10 by 10 grid with a number of black boxes that is greater than he grid size

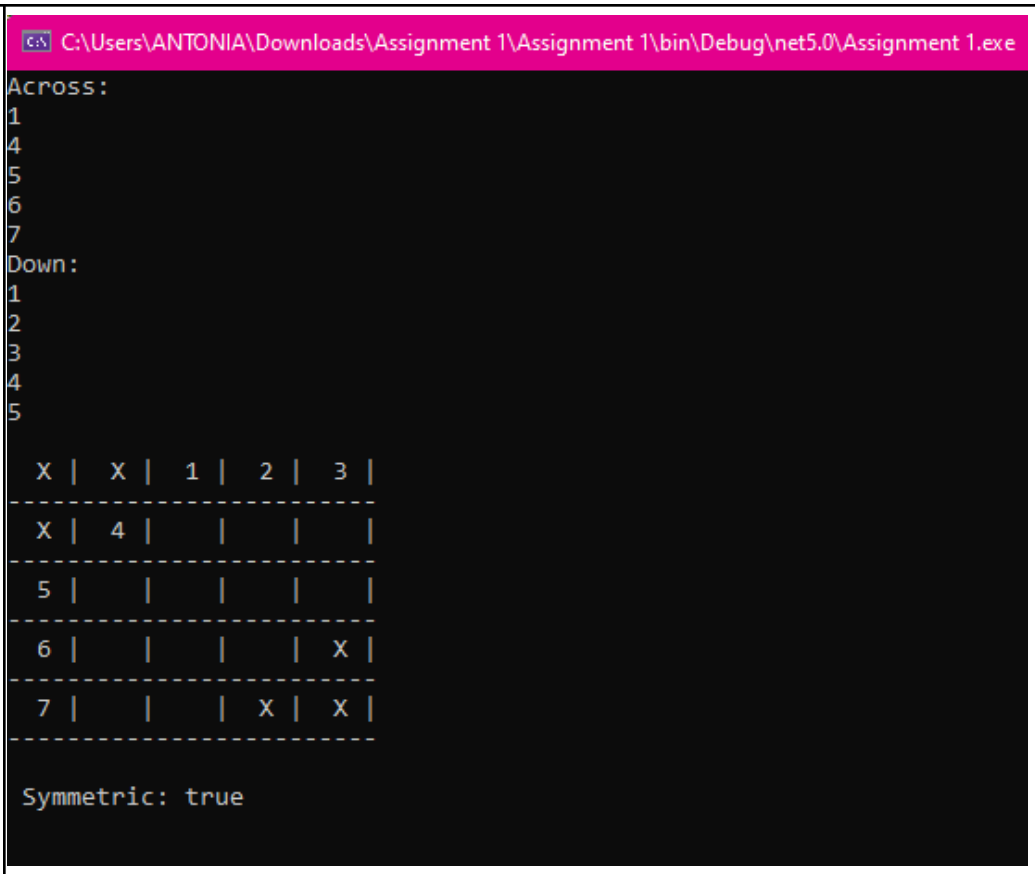
Main()	Puzzle puz = new Puzzle(10);
	puz.Initialize(101);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

Expected Output	An error message saying black boxes are more than grid size
Actual Output	

Test 6	
Description	Specifying grid boxes to get a symmetric grid in a 5 by 5 grid

Main() / Initialize()	*This code is inside Initialize() and the code that randomly asigns black squares is commented out*
	grid[0, 0].Color = TColor.BLACK;
	grid[0, 1].Color = TColor.BLACK;
	grid[1 ,0].Color = TColor.BLACK;
	grid[N - 1, N - 1].Colour = TColor.BLACK;
	grid[N - 1, N - 2].Color = TColor.BLACK;
	grid[N - 2, N - 1].Color = TColor.BLACK;
	
	* This code is inside Main()
	Puzzle puz = new Puzzle(5);
	puz.Initialize(0);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

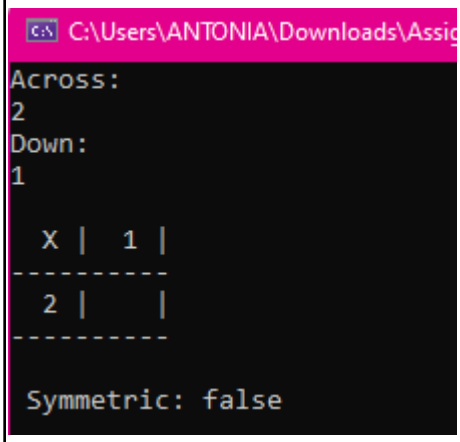
Expected Output	grid [0, 0], grid [0, 1], grid [1, 0], grid [4,4], grid [4, 3], grid [3, 4] are black squares and symmetric returns true.

Actual Output	

Test 7	
Description	Using a 2 by 2 grid

Main()	Puzzle puz = new Puzzle(2);
	puz.Initialize(1);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();

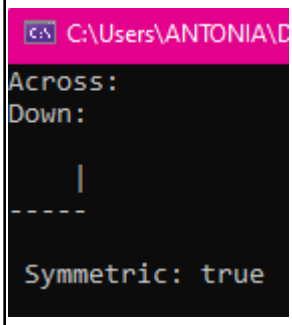
	bool s = puz.Symmetric();
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

Expected Output	A 2 by 2 grid with one black square
Actual Output	

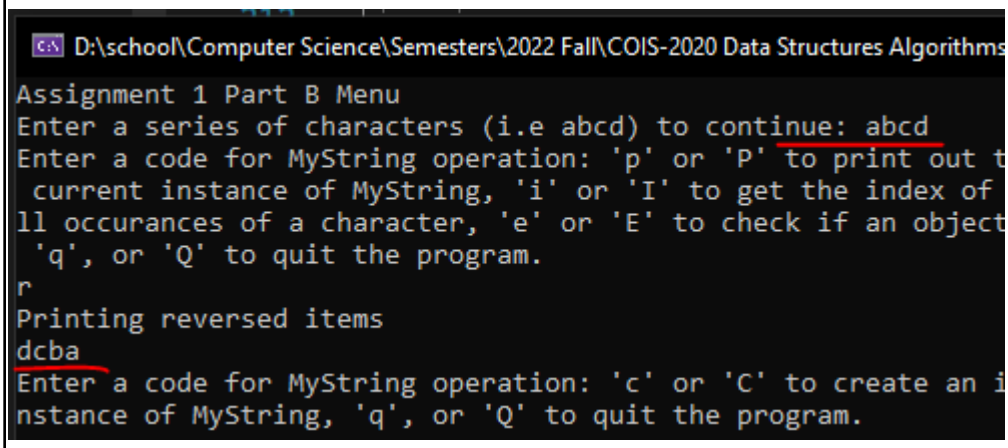
Test 8	
Description	Using a 1 by 1 grid

Main()	Puzzle puz = new Puzzle(1);
	puz.Initialize(0);
	puz.Number();
	puz.PrintClues();
	puz.PrintGrid();
	bool s = puz.Symmetric();

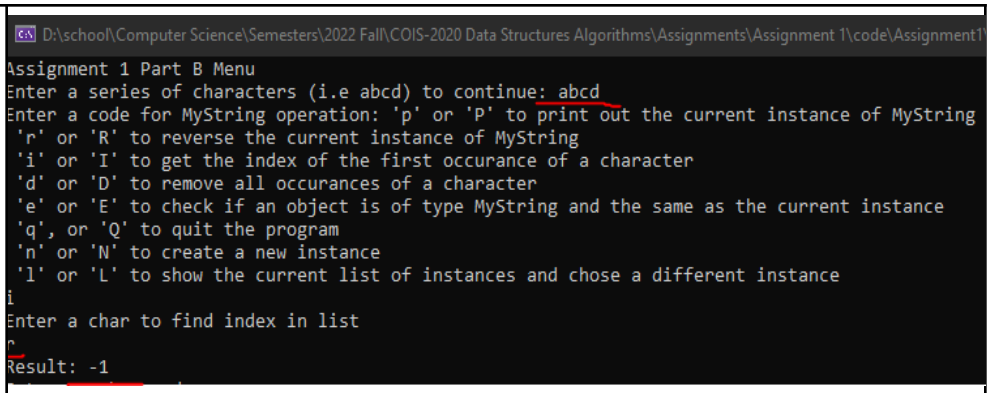
	if (s == true)
	Console.WriteLine("\n Symmetric: true");
	else
	Console.WriteLine("\n Symmetric: false");

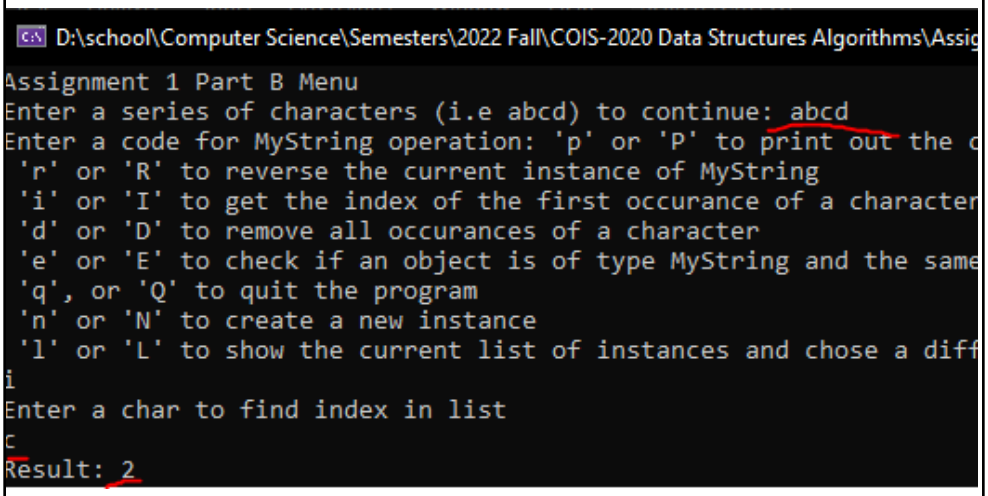
Expected Output	A 1 by 1 grid with no black boxes
Actual Output	

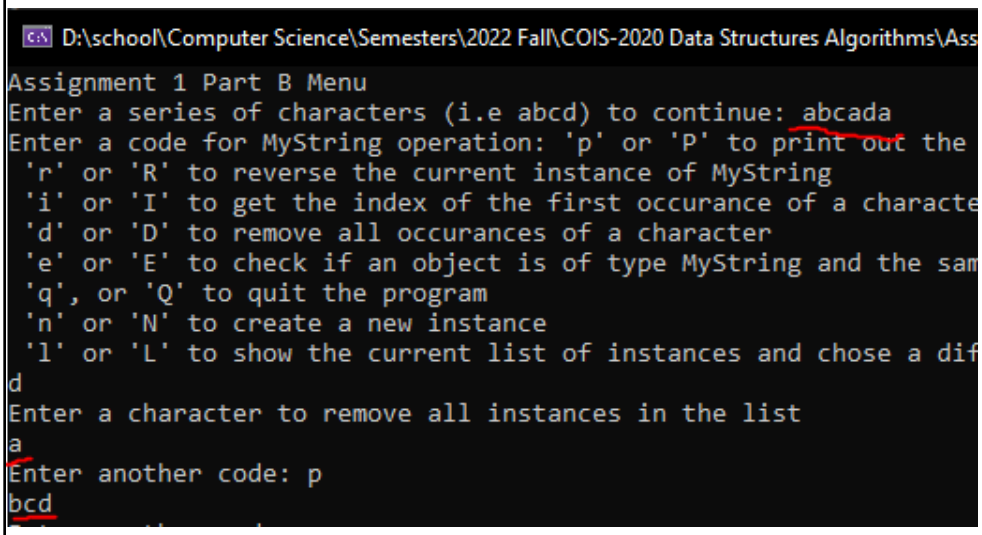
Part B - Tests

Test 1	
Description	Reverse the current instance of MyString
Input	abcd
Expected Output	dcba
Actual Output	

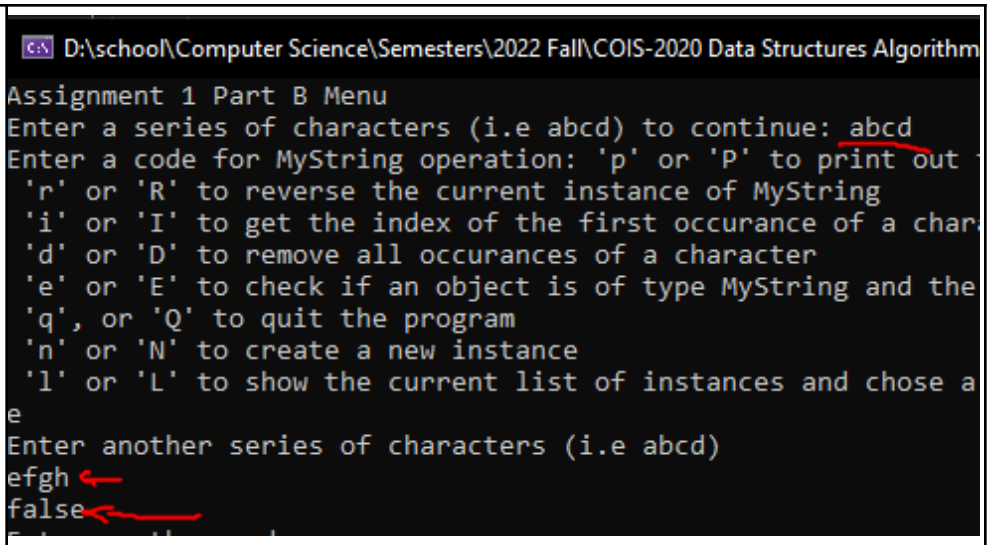
Test 2	
Description	Return the index of the first occurrence of c in this instance
Input	abcd - list input r - char c input
Expected Output	-1

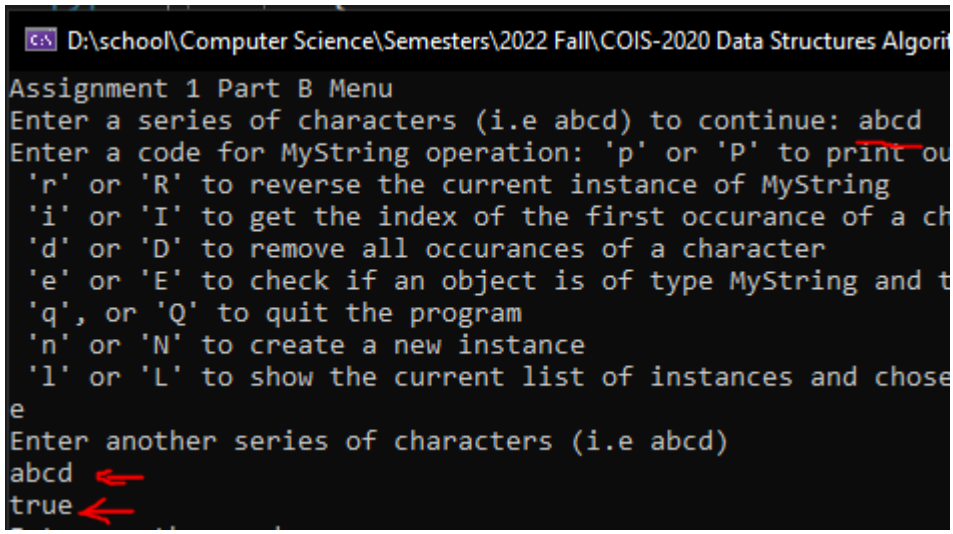
Actual Output	

Test 3	
Description	Return the index of the first occurrence of c in this instance
Input	abcd - list c - char c input
Expected Output	2
Actual Output	

Test 4	
Description	Remove all occurrences of c from this instance
Input	abcada - list input A - char c input
Expected Output	bcd
Actual Output	

Test 5	
Description	Return true if obj is both of type MyString and the same as this instance
Input	abcd - first instance input efgh - second instance input
Expected Output	false

Actual Output	

Test 6	
Description	Return true if obj is both of type MyString and the same as this instance
Input	abcd - first instance input abcd - second instance input
Expected Output	true
Actual Output	

Test 7	
Description	Print out this instance of MyString
Input	string - instance input
Expected Output	string
Actual Output