**Code Practice –**

* **Write a syntactically correct C# method definition called CountOdd.  The method should take an array of integers as it's parameter and should return the number of odd numbers in the array as a result.  For example, calling the function with an array containing 1, 3, 5, 4, 5, 2, 10, 7, 5, 7, 5 should return 8 because there are 8 odd numbers in the array.**
  + **private int CountOdd(int[] oddNumbers)**

**{**

**oddCount = 0;**

**foreach (int number in oddNumbers)**

**{**

**if (number % 2 != 0)**

**oddCount++**

**}**

**return oddCount;**

**}**

* **Assume that you've been asked to create an 5 by 5 TicTacToe application.  The board is represented by a 5 by 5 rectangular array ofcharacters.  Write a syntactically correct C# method definition called HasMatchingRow that can be used to determine if a player has a win in one of the rows on the board.  The method should take a rectangular array of characters, a row number and a character value as it's parameters and should return true or false as a result.  For example, calling the function with the following array, a row of 1 and O would return true.**
* Write a syntactically correct C# method definition called CountValues. The method should take an array of integers and an integer value as parameters and should return the number of times that the value occurs in the array as a result. For example, calling the function with an array containing 1, 3, 5, 4, 5, 2, 10, 7, 5, 7, 5 and the value 5 should return 4 because there are 4 5's in the array.
* **private int CountValues(int[] intArray, int searchValue)**

**{**

**count = 0;**

**foreach (int number in intArray)**

**{**

**if (number == searchValue)**

**count++;**

**}**

**return count;**

**}**

* Write a syntactically correct C# method definition called SumRow. The method should take a rectangular array of integers and a row number as it's parameters and should return the sum of the values in that row of the array as a result. For example, calling the function with the following array and a row of 0 would return 82+55+78 or 215.

|  |  |  |
| --- | --- | --- |
| 82 | 55 | 78 |
| 91 | 86 | 73 |
| 77 | 81 | 92 |

* **private int SumRow (int [,] rows, int row)**

**{**

**int sum = 0;**

**for (int col = 0; col < rows.GetLength(1); col++)**

**sum += rows[row, col];**

**return sum;**

**}**

* + - */\* To call this method:*

*int[,] rows = {{82, 55, 78}, {91, 86, 73}, {77, 81, 92}};*

*int sum = SumRows(rows, 0);*

*MessageBox.Show(sum.ToString()); \*/*

* **A palindrome is a word (or phrase) that reads the same forward as backward.  Examples of word palindromes in english are *kayak*, *racecar*, *redder*, *madam*, and *refer.***

**Write a syntactically correct C# method definition called IsPalindrome.  The method should take a string that represents a word as a parameter and should return true if the word is a palindrome and false if it is not.  For example:**

**madam is a palindrome**

**noon is a palindrome**

**apple is not a palindrome**

**You may assume that the parameter contains only lowercase characters, no spaces or punctuation.  HINT:  You wrote a function in**[**lab 2**](https://classes.lanecc.edu/mod/assign/view.php?id=1285003)**that "reverses" a string.  It may be useful in the solution of this problem.**

**FOR EXTRA CREDIT:**

**allow both upper and lower case characters but ignore case when determining if a word is a palindrome.**

**allow punctuation and/or spaces but consider only alpha characters when determining if a word is a palindrome.**

* **private bool IsPalindrome(string word)**

**{**

**bool palindrome = false;**

**palindrome = Reverse(word); //This isn't right... I wanted to reverse the word and check if it's equal to the original word.**

**if (word == palindrome)**

**return true;**

**return palindrome;**

**}**

**private string Reverse(string input)**

**{**

**string output = "";**

**for (int i = input.Length-1; i >= 0; i--)**

**output += input[i];**

**return output;**

**}**

* Eliza is a famous 1966 computer program written by Joseph Weizenbaum. It imitates a psychologist by rephrasing many of a patient's statements as questions and posing them to the patient.

Write a syntactically correct C# method definition called CreateElizaResponse. The method should take a string that represents the client's statement as a parameter and should return a string that represents an appropriate response. For example:

* + If the client's statement contains the word "my", (I am having trouble with my brother), respond with "Tell me more about your " followed by the noun in the statements (Tell me more about your brother).
  + If the client's statement contains the word love or hate, respond with "You have strong feelings about that!"
  + If neither of the above responses is appropriate, respond with one of the following "Please go on.", "Tell me more" or "Continue".
    - **private string CreateElizaResponse (string statement)**

**{**

**string output = “”;**

**if (statement.Contains(“love”) || statement.Contains(“hate”))**

**output = “You have strong feelings about that!”;**

**else if (statement.Contains(“my ”))**

**{**

**string [] words = statement.Split(null);**

**int index = Array.IndexOf(words, “my “);**

**output = “Tell me more about your “ + words[index+1];**

**}**

**else**

**output = “Continue”;**

**return output;**

**}**

* */\* Refer to Lab 2 \*/*
* private void card\_Click(**object sender**, EventArgs e)

{

PictureBox card = **(PictureBox)sender**;

int cardIndex = int.Parse**(card.Name.Substring(4))**;

if (index1 == NOT\_PICKED\_YET)

{

**index1 = cardIndex**;

card.Image = Image.FromFile(System.Environment.CurrentDirectory +

"\\Cards\\" + **cards[cardIndex];**

card.Enabled = false;

}

else

{

index2 = cardIndex;

LoadCard(cardIndex);

DisableAllCards();

flipTimer.Enabled = true;

}

}

* **object sender** = The first the object that generated the event; in this case a picture box… What did the user click?

**(PictureBox)sender** = casting “sender” as a PictureBox, because sender is the most generic object.

**(card.Name.Substring(4))** = using the substring method to get the character in index 4; gets the value in the index.

**Index1 = cardIndex** = assignment statement, assigning “cardIndex” to “index1.”

**cards[cardIndex]** = Going to array of filenames and getting the filename that matches with the PictureBox, and creating an image that gets shown in that PictureBox