Palindrome Detection Project

Task

Create a console application that takes a user-supplied input string and determines if it is a Palindrome.

Background

A palindrome is a string of letters and/or possibly numbers which read the same backwards and forwards. Spaces and punctuation are ignored, as are case differences. There are several famous and amusing ones:

A man, a plan, a canal: PANAMA!

Madam, in Eden, I'm Adam

Straw? No, too stupid a fad; I put soot on warts.

Do geese see God?

A nut for a jar of tuna.

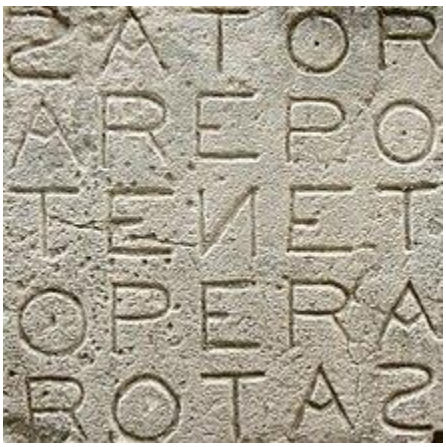
Al lets Della call Ed "Stella."

Step on no pets

Cigar? Toss it in a can. It is so tragic.

Go hang a salami! I'm a lasagna hog.

Sit on a potato pan, Otis



The concept dates back to the Roman Empire, with the oldest known palindrome also being an acrostic arranged in a square so that it reads the same from any of four directions! In Latin, it reads: "*sator arepo tenet opera rotas*", which means, "The sower Arepo holds with effort the wheels".

Instructions

Create a console app in Visual Studio.

Prompt the user to enter a string, and read the input text to a string **inputText**

Use a do-while loop which will repeat as long as the user keeps entering new potential palindromes. At the end of the loop, prompt the user to enter another or to just hit {ENTER} by itself to quit. If the user hits {ENTER} without entering any text, the variable will be the empty string, in which case, the program will terminate. Otherwise, the loop will go back to test the new text.

Remember that a string is actually a character array. For example, in: `string text = "CHOCOLATE";`

The individual letters can be referenced as members of the array, like so:

C	H	O	C	O	L	A	T	E
0	1	2	3	4	5	6	7	8

```
Console.WriteLine(text[0]); // C
Console.WriteLine(text[1]); // H
```

Inside the do-while loop, create a for-loop to iterate through the letters in the palindrome candidate.

Method 1

One way to solve this problem is to use a for-loop to filter out the spaces and punctuation of the string. Create a string **filteredInputText** and add the characters of the string to it, one at a time, after testing to verify that they are letters or numbers:

```
if (char.IsLetterOrDigit(inputText[index]))
{
        // implement code to add to filteredInputText
}
```

Then use another for-loop to go *backwards* by *decrementing* from the last index to the first, building a new reversed string. Then compare the two strings, case-insensitive.

Method 2

Another way to solve this problem is to have for-loop with complex initialization and update statements, moving pointers from both the beginning and end of the string *at the same time*, meeting in the middle. Ignore characters that aren't numbers or letters.

Your for-loop will initialize two counters: `int start=0, end=inputText.Length-1`

It will need to update both counters: `start++, end--`

The conditional must stop the loop when they meet: `start < end`

Rather than creating a filtered string, just skip characters that are not letters or numbers by putting two while loops inside the for-loop:

```
while (!char.IsLetterOrDigit(inputText[start]))
{
    start++;
}
while (!char.IsLetterOrDigit(inputText[end]))
{
    end--;
}
```

Since the while loops are also moving the pointers, you need to check again that start < end and **break** out of the loop if it isn't.

Finally, you need to test that the two characters are equal. Unlike string, there is no case-insensitive comparer for characters, so you have to cast them both ToLower() (or both to upper, if you prefer). You will need to use the static method inside the char class:

```
char.ToLower(inputText[start]).Equals(char.ToLower(inputText[end]));
```

Create a bool variable **isPalindrome** *inside* your do-loop but before the for-loop and set it equal to true. Then use boolean logic to continually "AND" your results together for each character:

```
isPalindrome = isPalindrome && // implement code here to check if characters are equal
```

With "AND" logic, the statement will only be true if both sides of the && are true. Continually running this inside the loop will ensure that if even one character doesn't match, isPalindrome will be set to false.

Example of the Output of a Finished Project