Week 2 – Debugging Questions

1. Fix the code to calculate remainder sum - 2

Previous

Next

Joe has two integers `a` and `b`. He wants to find the remainder when `a` is divided by `b` and do the following:

1. If the remainder is an even number greater than 2, then output the sum of `a`, `b`, and the remainder.
2. If the remainder is an odd number, then output the product of `a` and the remainder.
3. If the remainder is 0, 1, or 2, then output the sum of `a` and the remainder.

Joe has written a function `buggyRemainderOperation` that accepts integers `a` and `b` in its arguments. However, there are some bugs in the function logic. Your task is to fix buggyRemainderOperation to help Joe complete his task.

Function description

Complete the ***buggyRemainderOperation*** function in the editor below. It has the following parameter(s):

Name	Type	Description
a	INTEGER	The value a described in the problem.
b	INTEGER	The value b described in the problem.
Return	The function must return an *INTEGER* denoting the answer	

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
55 594	3025	If you input a = 55 and b = 594, the remainder when a is divided by b is 55 because 55 is less than 594. The

Input	Output	Output Description
		conditions in the function dictate that if the remainder is odd (which 55 is), the function should output the product of a and the remainder. Therefore, the output for the inputs 55 and 594 would be 55 * 55 = 3025.
625 779	390625	If you input a = 625 and b = 779, the remainder when a is divided by b is 625 because 625 is less than 779. The conditions in the function dictate that if the remainder is odd (which 625 is), the function should output the product of a and the remainder. Therefore, the output for the inputs 625 and 779 would be 625 * 625 = 390625.
832 281	1383	If you input a = 832 and b = 281, the remainder when a is divided by b is 270 because 832 % 281 = 270. The conditions in the function dictate that if the remainder is an even number greater than 2 (which 270 is), the function should output the sum of a, b and the remainder. Therefore, the output for the inputs 832 and 281 would be 832 + 281 + 270 = 1383.

```
static int buggyRemainderOperation(int a, int b) {
        // Fix the code here
        int remainder = a / b;
        if (remainder > 2 && remainder % 2 == 0) {
            return a + b + remainder;
        } else if (remainder % 2 == 1) {
            return a * remainder;
        } else if (remainder >= 0 && remainder <= 2) {
            return a + remainder;
        } else {
            Console.WriteLine("Unexpected case");
            return -1;  // Return some error code
        }
```

```
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggyRemainderOperation(int a, int b) {
        // Fix the code here
        int remainder = a % b;
        if (remainder > 2 && remainder % 2 == 0) {
            return a + b + remainder;
        } else if (remainder % 2 != 0) {
            return a * remainder;
        } else if (remainder >= 0 && remainder <= 2) {
            return a + remainder;
        } else {
            Console.WriteLine("Unexpected case");
            return -1;  // Return some error code
        }
    }

    static void Main(string[] args) {
        int a = Int32.Parse(Console.ReadLine().Trim());

        int b = Int32.Parse(Console.ReadLine().Trim());
        int result = buggyRemainderOperation(a, b);

        Console.WriteLine(result);
    }
}
```

2. Fix the code to perform character replication in string - 2

Previous
Next

Bob has a string `S` of size `N`. He wants to create a transformed string by replicating the first vowel character in `S` across the entire length of `S`. If no vowel is present, he wants to replicate the first character of `S`.

Bob has written a function `ReplicateVowel` that accepts the integer `N` and string `S` in its arguments. However, there are some bugs in the function logic. Your task is to fix ReplicateVowel to help Bob complete his task.

Function description

Complete the ***ReplicateVowel*** function in the editor below. It has the following parameter(s):

Name	Type	Description
N	INTEGER	The size of s.
S	STRING	The given string.
Return	The function must return a *STRING* denoting the transformed string S.	

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
3 Bob	ooo	The length of bob is 3. The first vowel is 'o' so the output is "ooo".
5 sugar	uuuuu	The length of sugar is 5. The first vowel is 'u', so the output is "uuuuu".
5 Hello	eeeee	The length of hello is 5. The first vowel is 'e', so the output is "eeeee".

```
static string ReplicateVowel(int N, string S) {
        //Fix the code here
        string vowels = "aeiouAEIOU";
        foreach(char c in S) {
            if(!vowels.Contains(c)) {
                return new String(c, N);
            }
        }
        return new String(S[0], N);
```

```
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static string ReplicateVowel(int N, string S) {
        //Fix the code here
        string vowels = "aeiouAEIOU";
        foreach(char c in S) {
            if(vowels.Contains(c)) {
                return new String(c, N);
            }
        }
        return new String(S[0], N);
    }

    static void Main(string[] args) {
        int N = Int32.Parse(Console.ReadLine().Trim());

        string S = Console.ReadLine();
        String result = ReplicateVowel(N, S);

        Console.WriteLine(result);
    }
}
```

3. Fix the code to replace vowels in a string - 2

Previous
Next

Bob has a string `s`. He wants to replace all vowels in `s` with the number 3. However, if a vowel is surrounded by consonants on both sides, it should be replaced with the number 5 instead.

Bob has written a function `BuggyReplaceVowels` that accepts the string `S` in its arguments and returns the updated version of the string `s`. However, there are some bugs in the function logic.

Your task is to fix the issues in BuggyReplaceVowels to fix the code.

Function description
Complete the ***BuggyReplaceVowels*** function in the editor below. It has the following parameter(s):

Name	Type	Description
s	STRING	The given string
Return	The function must return a *STRING* denoting the replacing vowels with 5 if surrounded by consonants and with 3 if not.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
`hello`	`h5ll3`	'e' is surrounded by consonants. However, 'o' is not surrounded. Therefore the transformed string is h5ll3.
`world`	`w5rld`	'o' is surrounded by consonants. Hence the answer is w5rld.
`programming`	`pr5gr5mm5ng`	'o' and 'a' and 'i' are surrounded by consonants. Hence, the answer is pr5gr5mm5ng after replacing all 'o' in the string.

```
public static string BuggyReplaceVowels(string s)
{
    // Fix the code here
    char[] result = new char[s.Length];

    for (int i = 0; i < s.Length; i++)
    {
        char ch = s[i];
        char lowerCh = char.ToLower(ch);

        if (lowerCh == 'a' || lowerCh == 'e' || lowerCh == 'i' || lowerCh == 'o' || lowerCh =
= 'u')
        {
            if (i > 0 && i < s.Length - 1 && !char.IsWhiteSpace(s[i - 1]) && !char.IsWhiteSpa
ce(s[i + 1]) && !(char.ToLower(s[i - 1]) == 'a' || char.ToLower(s[i - 1]) == 'e' || char.ToLo
wer(s[i - 1]) == 'i' || char.ToLower(s[i - 1]) == 'o' || char.ToLower(s[i - 1]) == 'u') && !(
char.ToLower(s[i + 1]) == 'a' || char.ToLower(s[i + 1]) == 'e' || char.ToLower(s[i + 1]) == '
i' || char.ToLower(s[i + 1]) == 'o' || char.ToLower(s[i + 1]) == 'u'))
            {
                result[i] = '3';
            }
            else
            {
                result[i] = '5';
            }
        }
        else
        {
            result[i] = ch;
        }
    }

    return new string(result);
}
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;
```

```
class Solution {
    public static string BuggyReplaceVowels(string s)
{
    // Fix the code here
    char[] result = new char[s.Length];

    for (int i = 0; i < s.Length; i++)
    {
        char ch = s[i];
        char lowerCh = char.ToLower(ch);

        if (lowerCh == 'a' || lowerCh == 'e' || lowerCh == 'i' || lowerCh == 'o' || lowerCh =
= 'u')
        {
            if (i > 0 && i < s.Length - 1 && !char.IsWhiteSpace(s[i - 1]) && !char.IsWhiteSpa
ce(s[i + 1]) && !(char.ToLower(s[i - 1]) == 'a' || char.ToLower(s[i - 1]) == 'e' || char.ToLo
wer(s[i - 1]) == 'i' || char.ToLower(s[i - 1]) == 'o' || char.ToLower(s[i - 1]) == 'u') && !(
char.ToLower(s[i + 1]) == 'a' || char.ToLower(s[i + 1]) == 'e' || char.ToLower(s[i + 1]) == '
i' || char.ToLower(s[i + 1]) == 'o' || char.ToLower(s[i + 1]) == 'u'))
            {
                result[i] = '5';

            }
            else
            {
                result[i] = '3';
            }
        }
        else
        {
            result[i] = ch;
        }
    }

    return new string(result);
}

    static void Main(string[] args) {
        string s = Console.ReadLine();
        String result = BuggyReplaceVowels(s);

        Console.WriteLine(result);
    }
}
```

4. Fix the code to find sum of even numbers - 2

John has an array `arr` of size `n`. He wants to find the sum of squares of all even numbers in `arr` that are also divisible by 4.

John has written a function `buggySumOfEvenSquaresDivisibleByFour` that accepts `n` and `arr` as a list of integers in its arguments. However, there are some bugs in the function logic.

Your task is to fix the issues in buggySumOfEvenSquaresDivisibleByFour to help john complete the task.

Function description

Complete the ***buggySumOfEvenSquaresDivisibleByFour*** function in the editor below. It has the following parameter(s):

Name	Type	Description
arr	INTEGER ARRAY	The given array of integers.
Return	The function must return an *INTEGER* denoting the sum of squares of all even numbers that are divisible by 4	

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
10 1 2 3 4 5 6 7 8 9 10	80	4 and 8 satisfy the requirements. Hence, the answer is 16 + 64 = 80.
4 4 4 4 4	64	All of the elements in arr satisfy the given requirements. Hence, the answer is 16+16+16+16=64.

Input	Output	Output Description
4 1 1 1 1	0	No element in arr satisfies the given requirement so the answer is 0.

```
static int buggySumOfEvenSquaresDivisibleByFour(List<int> arr) {
        // Fix the code here
        int result = 0;

        foreach (int num in arr)
        {
            if (num / 2 == 0 && num % 4 == 0)
            {
                result += num * num;
            }
        }

        return result;

    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggySumOfEvenSquaresDivisibleByFour(List<int> arr) {
        // Fix the code here
        int result = 0;

        foreach (int num in arr)
```

```
        {
            if (num % 2 == 0 && num % 4 == 0)
            {
                result += num * num;
            }
        }

        return result;

    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<int> arr = new List<int>();
        for(int j=0; j<n; j++) {
            arr.Add(Int32.Parse(Console.ReadLine().Trim()));
        }
        int result = buggySumOfEvenSquaresDivisibleByFour(arr);

        Console.WriteLine(result);
    }
}
```

5. Fix the code to count the occurrences of a target character - 2

Previous

Next

Jake has a string array `arr` of size `n` and two special characters, `target1` and `target2`. He wants to count the total number of occurrences of `target1` in all strings in `arr`, but only if `target1` is immediately followed by `target2` in the string.

Jake has written a function `buggyCountTargetCharacterPairs` that accepts `arr`, `target1`, and `target2` in its arguments. However, there are some bugs in the function logic. Your task is to fix the issues in `buggyCountTargetCharacterPairs` to help Jake complete his task.

Function description

Complete the ***buggyCountTargetCharacterPairs*** function in the editor below. It has the following parameter(s):

Name	Type	Description
arr	STRING ARRAY	The given array of strings.

Name	Type	Description
target1	CHARACTER	The character target1 described in the problem.
target2	CHARACTER	The character target2 described in the problem.
Return	The function must return an *INTEGER* denoting the total count of occurrences of target1 when it is followed by target2 in all strings in arr.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
3 abc def gab a b	2	'a' followed by 'b' appears twice in abc and in gab. Thus, the answer is 2.
3 aaa aaa aaa a a	6	'a' followed by 'a' appears twice in aaa. We have 3 strings of "aaa" in the given array. Hence the answer is 3*2=6.
3 abc def ghi d e	1	d followed by e appears only once in the second string def. Hence the answer is 1.

```
// Fix the code here
        int count = 0;
        foreach (var s in arr)
        {
            for (int i = 0; i < s.Length - 1; i++)
            {
                if (s[i] == target1 || s[i + 1] == target2)
                {
                    count++;
                }
            }
        }
        return count;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static int buggyCountTargetCharacterPairs(List<string> arr, char target1, char tar
get2)
    {
        // Fix the code here
        int count = 0;
        foreach (var s in arr)
        {
            for (int i = 0; i < s.Length - 1; i++)
            {
                if (s[i] == target1 && s[i + 1] == target2)
                {
                    count++;
                }
            }
        }
        return count;
    }
```

```
    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<string> arr = new List<string>();
        for(int j=0; j<n; j++) {
            arr.Add(Console.ReadLine());
        }

        char target1 = char.Parse(Console.ReadLine());

        char target2 = char.Parse(Console.ReadLine());
        int result = buggyCountTargetCharacterPairs(arr, target1, target2);

        Console.WriteLine(result);
    }
}
```

6. Fix the code to check if a given string is a palindrome - 2

Previous
Next

You are given a partially implemented code to check if a given string is a specific type of palindrome known as a **"mirrored string"**. If the given string is a **mirrored string**, return 1; else return 0.

A **mirrored string** is a string that appears the same when looked at in a mirror (ignoring spaces, punctuation, and capitalization). This means that each character in the string must be symmetrical along the vertical axis and the string should read the same backwards as forwards. For instance, the string "Aibohphobia" is a mirrored string as each character appears the same in a mirror and the string reads the same backwards.

The code reads a string and calls the function `buggyIsMirroredString` to determine if the string is a **mirrored string**.

However, the function contains a bug that causes it to produce incorrect results. Your task is to find and fix the bug in the `buggyIsMirroredString` function.

Function description

Complete the ***buggyIsMirroredString*** function in the editor below. It has the following parameter(s):

Name	Type	Description
str	STRING	The given string.

Return	The function must return an *INTEGER* denoting the value 1 if the given string is a mirrored string, else return 0.

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
Aibohphobia	1	This is a mirrored string.
WOW	1	This is a mirrored string.
Hello	0	This is not a mirrored string.

```
static int buggyIsMirroredString(String str) {
        // Fix the code here
        string mirrored_chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        str = str.ToLower();
        return (str == new string(str.ToCharArray().Reverse().ToArray()) &&
                str.All(ch => mirrored_chars.Contains(ch))) ? 1 : 0;
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggyIsMirroredString(String str) {
        // Fix the code here
        string mirrored_chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        str = str.ToUpper();
        return (str == new string(str.ToCharArray().Reverse().ToArray()) &&
```

```
                str.All(ch => mirrored_chars.Contains(ch))) ? 1 : 0;
    }

    static void Main(string[] args) {
        string str = Console.ReadLine();
        int result = buggyIsMirroredString(str);

        Console.WriteLine(result);
    }
}
```

7. Fix the code to determine if a given string is a valid email address - 2

Previous
Next

You are given a partially implemented code to determine if a given string is a valid email address. The code reads a string and calls the function `buggyIsValidEmail` to check if the string is a valid email address.

However, the function contains an issue. Your task is to find and fix the bug in the `buggyIsValidEmail` function.

Notes:
A valid email address for this task should:

1. Consist of a local part followed by an "@" symbol, and a domain part.
2. The local part may contain alphanumeric characters (both uppercase and lowercase), periods (.), and plus signs (+), but it should not start or end with a period or plus sign.
3. The domain part may contain alphanumeric characters (both uppercase and lowercase) and hyphens (-), but it must start and end with an alphanumeric character.
4. The domain part should also contain at least one period (.) symbol, indicating a domain suffix like .com, .org, etc.

Thus, if the input string is "john.doe@example.com", the output should be true, but if the input string is "john..doe@example.com" or ".john@example.com", the output should be false because two consecutive periods are not allowed and the local part shouldn't start with a period.
Function description
Complete the ***buggyIsValidEmail*** function in the editor below. It has the following parameter(s):

Name	Type	Description
emailStr	STRING	The given email address.

Return	The function must return a *BOOLEAN* denoting the value 1 if it is a valid email address and 0 if it is not a valid email address.

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
john.doe@example.com	1	This is a valid email address.
john..doe@example.com	0	Two dots are present in the given email. Hence, it is not a valid email address.
.john@example.com	0	A dot is present in the beginning of the given email. Hence, it is not a valid email address.

```
// fix the code here
        int atPos = emailStr.IndexOf('@');

        if (atPos == -1)
        {
            return false;
        }

        string local = emailStr.Substring(0, atPos);
        string domain = emailStr.Substring(atPos + 1);

        if (local[0] == '.' || local[0] == '+' || local[^1] == '.' || local[^1] == '+' || !lo
cal.Contains(".."))
        {
            return false;
        }
```

```
        if (domain[0] == '-' || domain[0] == '.' || domain[^1] == '-
' || domain[^1] == '.' || !domain.Contains(".."))
        {
            return false;
        }

        if (!domain.Contains('.'))
        {
            return false;
        }

        return true;
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static bool BuggyIsValidEmail(string emailStr)
    {
        // fix the code here
        int atPos = emailStr.IndexOf('@');

        if (atPos == -1)
        {
            return false;
        }

        string local = emailStr.Substring(0, atPos);
        string domain = emailStr.Substring(atPos + 1);

        if (local.StartsWith('.') || local.StartsWith('+') || local.EndsWith('.') || local.En
dsWith('+') || local.Contains(".."))
        {
            return false;
        }
```

```
        if (domain.StartsWith('.') || domain.StartsWith('+') || domain.EndsWith('.') || domai
n.EndsWith('+') || domain.Contains(".."))
        {
            return false;
        }

        if (!domain.Contains('.'))
        {
            return false;
        }

        return true;
    }

    static void Main(string[] args) {
        string emailStr = Console.ReadLine();
        bool result = BuggyIsValidEmail(emailStr);

        Console.WriteLine(result ? 1: 0);
    }
}
```

8. Fix the code to determine if a given string is a valid URL - 2

Previous
Next

You are given a partially implemented code to determine if a given string is a valid URL with a specific domain suffix. The code reads a string and calls the function `buggyIsValidURLWithSpecificSuffix` to check if the string is a valid URL and ends with the domain suffix ".org".

However, the function contains a bug. Your task is to find and fix the bug in the `buggyIsValidURLWithSpecificSuffix` function.

Notes:
A valid URL should:

1. Start with a valid scheme (e.g., "http", "https", "ftp").
2. Followed by "://".
3. Then, a domain part, which may contain alphanumeric characters (both uppercase and lowercase), hyphens (-), and periods (.). The domain part must start and end with an alphanumeric character.
4. End with the domain suffix ".org".
5. Optionally, a path part after the domain, which may contain alphanumeric characters (both uppercase and lowercase), hyphens (-), slashes (/), and periods (.)

For example, if the input string is "https://www.example.org/test", the output should be true, but if the input string is "https://www.example.com/test" or "http:/www.example.org/test", the output should be false because the URL does not have a valid scheme or the correct domain suffix.

Function description

Complete the ***buggyIsValidURLWithSpecificSuffix*** function in the editor below. It has the following parameter(s):

Name	Type	Description
url	STRING	The given url.
Return	The function must return a *BOOLEAN* denoting the value 1 if it is a valid url according to given rules. Otherwise, return the value 0.	

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
`https://www.example.org/test`	`1`	This is a valid url.
`https://www.example.com/test`	`0`	This is not a valid url.
`http:/www.example.org/test`	`0`	This is not a valid url.

```
// Fix the code here
        var schemes = new[] {"http://", "https://", "ftp://"};
        var parts = url.Split(new[] { "://" }, StringSplitOptions.None);
        if (parts.Length != 2 || !schemes.Contains(parts[0])) {
            return false;
        }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
```

```
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static bool buggyIsValidURLWithSpecificSuffix(String url) {
        // Fix the code here
        var schemes = new[] {"http", "https", "ftp"};
        var parts = url.Split(new[] { "://" }, StringSplitOptions.None);
        if (parts.Length != 2 || !schemes.Contains(parts[0])) {
            return false;
        }
        parts = parts[1].Split(new[] { "/" }, StringSplitOptions.None);
        var domain = parts[0];
        if (!domain.EndsWith(".org") || !char.IsLetterOrDigit(domain[0]) || !char.IsLetterOrD
igit(domain[domain.Length - 5])) {
            return false;
        }
        if (domain.Any(c => !char.IsLetterOrDigit(c) && c != '-' && c != '.')) {
            return false;
        }
        foreach (var part in parts.Skip(1)) {
            if (part.Any(c => !char.IsLetterOrDigit(c) && c != '-
' && c != '/' && c != '.')) {
                return false;
            }
        }
        return true;
    }

    static void Main(string[] args) {
        string url = Console.ReadLine();
        bool result = buggyIsValidURLWithSpecificSuffix(url);

        Console.WriteLine(result ? 1: 0);
    }
}
```

9. Fix the code to find sum of squares of all even numbers - 2

John has an array `arr` of size `n`. He wants to find the sum of squares of all even numbers in `arr` that are at odd-indexed positions. It is given that arr is 0-indexed.

John has written a function `buggySumOfEvenSquaresAtOddIndices` that accepts `n` and `arr` as a list of integers in its arguments. However, there are some bugs in the function logic.

Your task is to fix the issues in `buggySumOfEvenSquaresAtOddIndices` to help John complete his task.

Function description

Complete the ***buggySumOfEvenSquaresAtOddIndices*** function in the editor below. It has the following parameter(s):

Name	Type	Description
n	INTEGER	The size of arr.
arr	INTEGER ARRAY	The given array.
Return	The function must return an *INTEGER* denoting the sum of squares of all even numbers in arr that are at odd-indexed positions. It is given that arr is 0-indexed.	

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
5 1 2 3 4 5	20	Only 4 (at index 3) is an even number at an odd-indexed position, 4^2 = 16, and 2 (at index 1) is also an even number at an odd-indexed position, 2^2 = 4, so total is 16 + 4 = 20.
3 2 4 6	16	Only 4 (at index 1) is an even number at an odd-indexed position.

Input	Output	Output Description
4 1 3 5 7	0	There are no even numbers at odd-indexed positions.

```
// Fix ther code here
        int sum = 0;
        for(int i = 1; i < n; i += 2) {
            if(arr[i] % 2 == 0) {
                sum += (int)Math.Pow(arr[i], 1);
            }
        }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggySumOfEvenSquaresAtOddIndices(int n, List<int> arr) {
        // Fix ther code here
        int sum = 0;
        for(int i = 0; i < n; i ++) {
            if(arr[i] % 2 == 0 && i % 2 != 0) {
                sum += (int)Math.Pow(arr[i], 2);
            }
        }
        return sum;
    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());
```

```
        List<int> arr = new List<int>();
        for(int j=0; j<n; j++) {
            arr.Add(Int32.Parse(Console.ReadLine().Trim()));
        }
        int result = buggySumOfEvenSquaresAtOddIndices(n, arr);

        Console.WriteLine(result);
    }
}
```

10. Fix the code to find sum of squares of all odd numbers - 2

Previous
Next

John has an array `arr` of size `n`. He wants to find the sum of squares of all odd numbers in `arr` that are also perfect squares.

John has written a function `buggySumOfOddPerfectSquareSquares` that accepts `n` and `arr` as a list of integers in its arguments. However, there are some bugs in the function logic.

Your task is to fix the issues in `buggySumOfOddPerfectSquareSquares` to help John complete his task.
Function description
Complete the ***buggySumOfOddPerfectSquareSquares*** function in the editor below. It has the following parameter(s):

Name	Type	Description
n	INTEGER	The size of arr.
arr	INTEGER ARRAY	The given array.
Return	The function must return an *INTEGER* denoting the sum of squares of all odd numbers in arr that are also perfect squares.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
5 1 4 9 16 25	707	The answer here is 9*9 + 25*25 + 1*1 = 707.
5 2 4 6 8 10	0	There are no odd numbers in the array. Hence, the answer is 0.
5 1 3 5 7 9	82	The answer here is 9*9 + 1*1 = 82.

```
// Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++)
        {
            double root = Math.Sqrt(arr[i]);
            if (arr[i] % 2 != 0 && (int)root * (int)root != arr[i])
            {
                sum += arr[i] * arr[i];
            }
        }
        return sum;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
```

```
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static int buggySumOfOddPerfectSquareSquares(int n, List<int> arr)
    {
        // Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++)
        {
            double root = Math.Sqrt(arr[i]);
            if (arr[i] % 2 != 0 && (int)root * (int)root == arr[i])
            {
                sum += arr[i] * arr[i];
            }
        }
        return sum;
    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<int> arr = new List<int>();
        for(int j=0; j<n; j++) {
            arr.Add(Int32.Parse(Console.ReadLine().Trim()));
        }
        int result = buggySumOfOddPerfectSquareSquares(n, arr);

        Console.WriteLine(result);
    }
}
```

11. Fix the code to determine if a given string contains all unique characters - 2

Previous
Next

You are given a partially implemented code to determine if a given string contains all unique characters, excluding whitespace. The code reads a string and calls the function `buggyHasUniqueNonWhitespaceCharacters` to check if the string, after removing all whitespace, has all unique characters.

However, the function contains a bug that causes it to produce incorrect results. Your task is to find and fix the bug in the `buggyHasUniqueNonWhitespaceCharacters` function.

Function description

Complete the ***buggyHasUniqueNonWhitespaceCharacters*** function in the editor below. It has the following parameter(s):

Name	Type	Description
str	STRING	The given string.
Return	The function must return a *BOOLEAN* denoting the value 1 if the string, after removing all whitespace, has all unique characters. Otherwise, we return the value 0.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
`hello world`	`0`	The answer is False, since 'l' and 'o' repeated. Hence we return 0.
`123 456 789`	`1`	The answer is True, since all characters are unique. Hence we return 1.
`a b c d e f`	`1`	The answer is True, since all characters are unique. Hence the answer is 1.

```
// Fix the code here
        str = str.Replace(" ", ".");
        return str.Distinct().Count() == str.Length;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
```

```
using System.Text.RegularExpressions;
using System;


class Solution {
    public static bool buggyHasUniqueNonWhitespaceCharacters(string str) {
        // Fix the code here
        str = str.Replace(" ", "");
        return str.Distinct().Count() == str.Length;
    }

    static void Main(string[] args) {
        string str = Console.ReadLine();
        bool result = buggyHasUniqueNonWhitespaceCharacters(str);

        Console.WriteLine(result ? 1: 0);
    }
}
```

12. Fix the code to find number of words in a sentence that start with a specified character - 2

Previous
Next

Jill has a string `str`. She wants to find the number of words in `str` that both start and end with a specified character `c`.

Jill has written a function `buggyCountWordsStartingAndEndingWithCharacter` that accepts `str` and `c` in its arguments. However, there are some bugs in the function logic. Fix the issues in `buggyCountWordsStartingAndEndingWithCharacter` to help Jill complete her task.

Notes:

- It is guaranteed that all inputs are provided in lowercase.

Function description
Complete the ***buggyCountWordsStartingAndEndingWithCharacter*** function in the editor below. It has the following parameter(s):

Name	Type	Description
str	STRING	The given string.

Name	Type	Description
c	CHARACTER	The character c described in the problem.
Return		The function must return an *INTEGER* denoting the number of words in str that both start and end with a specified character c.

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
oreo world h	0	No word starts and end with 'h'. Hence answer is 0.
cat bat matm cat m	1	matm starts and ends with 'm'. Thus, answer is 1.
apple applea apple a	1	applea starts and ends with 'a'. Hence answer is 1.

```
// Fix the code here
        int count = 0;
        string[] words = str.Split(' ');
        foreach(string word in words) {
            if(word.StartsWith(c.ToString()) || word.EndsWith(c.ToString())) {
                count++;
            }
        }
        return count;
```

Ans:-

```
using System.CodeDom.Compiler;
```

```
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static int buggyCountWordsStartingAndEndingWithCharacter(String str, char c) {
        // Fix the code here
        int count = 0;
        string[] words = str.Split(' ');
        foreach(string word in words) {
            if(word.StartsWith(c.ToString()) && word.EndsWith(c.ToString())) {
                count++;
            }
        }
        return count;
    }

    static void Main(string[] args) {
        string str = Console.ReadLine();

        char c = char.Parse(Console.ReadLine());
        int result = buggyCountWordsStartingAndEndingWithCharacter(str, c);

        Console.WriteLine(result);
    }
}
```

13. Fix the code to count the number of strings in a list that have a specified length - 2

Previous
Next

Jake has a string array `arr` of size `n`, an integer value `length`, and a character `targetChar`. He wants to count the total number of strings in `arr` whose length is equal to the given integer value and that also start with the `targetChar`.

Jake has written a function `buggyCountStringsOfTargetLengthAndChar` that accepts `arr`, `length`, and `targetChar` in its arguments. However, there are some bugs in the function logic.

Your task is to fix the issues in `buggyCountStringsOfTargetLengthAndChar` to help Jake complete his task.
Function description

Complete the ***buggyCountStringsOfTargetLengthAndChar*** function in the editor below. It has the following parameter(s):

Name	Type	Description
n	INTEGER	The size of arr.
arr	STRING ARRAY	The given string array.
length	INTEGER	The value length described in the problem.
targetChar	CHARACTER	The value targetChar described in the problem.
Return	The function must return an *INTEGER* denoting the count the total number of strings in arr whose length is equal to the given integer value and start with the targetChar.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
4 apple pear banana apricot 5 a	1	Only apple starts with 'a' and has length of 5.
3 cat	2	cat and cow, both start with 'c' and have length of 3.

Input	Output	Output Description
dog cow 3 c		
5 zebra lion tiger elephant monkey 5 t	1	Only tiger starts with t and has length of 5.

```
// Fix the code here
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i].Length == length && arr[i][1] == targetChar) {
                count++;
            }
        }
        return count;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static int buggyCountStringsOfTargetLengthAndChar(int n, List<string> arr, int len
gth, char targetChar) {
        // Fix the code here
```

```
        int count = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i].Length == length && arr[i][0] == targetChar) {
                count++;
            }
        }
        return count;
    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<string> arr = new List<string>();
        for(int j=0; j<n; j++) {
            arr.Add(Console.ReadLine());
        }

        int length = Int32.Parse(Console.ReadLine().Trim());

        char targetChar = char.Parse(Console.ReadLine());
        int result = buggyCountStringsOfTargetLengthAndChar(n, arr, length, targetChar);

        Console.WriteLine(result);
    }
}
```

14. Fix the code to find product of prime numbers in a list - 2

Previous
Next

Sam has an array `arr` containing `n` integers. He wants to calculate the product of all prime numbers in `arr` that are greater than 10. Therefore, he has written a function `buggyProductOfPrimesGreaterThanTen` that accepts `n` and stores `arr` as a list of integers as its arguments.

However, there are some issues in the code for this function. Fix the issues in `buggyProductOfPrimesGreaterThanTen` to help Sam complete his task.

Function description

Complete the ***buggyProductOfPrimesGreaterThanTen*** function in the editor below. It has the following parameter(s):

Name	Type	Description
n	INTEGER	The size of arr.

Name	Type	Description
arr	INTEGER ARRAY	The given array.
Return	The function must return an *INTEGER* denoting the product of all prime numbers in arr that are greater than 10.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
5 11 12 13 14 15	143	The two prime numbers in arr are 11 and 13. Hence, 11 * 13 = 143 is the answer.
7 10 11 12 13 14 15 16	143	The two prime numbers in arr are 11 and 13. Hence, 11 * 13 = 143 is the answer.
3 11 13 17	2431	The three prime numbers in arr are 11, 13 and 17. Hence, 11*13*17 = 2431 is the answer.

```
public static int buggyProductOfPrimesGreaterThanTen(int n, List<int> arr) {
        // Fix the code here
        int product = 1;
        foreach(var num in arr) {
            if (num < 10 && IsPrime(num)) {
```

```
                product *= num;
            }
        }
        return product;
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static bool IsPrime(int num) {
        if (num <= 1) return false;
        if (num <= 3) return true;
        if (num % 2 == 0 || num % 3 == 0) return false;
        int i = 5;
        while (i * i <= num) {
            if (num % i == 0 || num % (i + 2) == 0) return false;
            i += 6;
        }
        return true;
    }

    public static int buggyProductOfPrimesGreaterThanTen(int n, List<int> arr) {
        // Fix the code here
        int product = 1;
        foreach(var num in arr) {
            if (num > 2 && IsPrime(num)) {
                product *= num;
            }
        }
        return product;
    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<int> arr = new List<int>();
```

```
        for(int j=0; j<n; j++) {
            arr.Add(Int32.Parse(Console.ReadLine().Trim()));
        }
        int result = buggyProductOfPrimesGreaterThanTen(n, arr);

        Console.WriteLine(result);
    }
}
```

15. Fix the code to find the number of times a target substring appears in a given string - 2

Previous
Next

Jack has two strings, `str` and `t`. He wants to find the total number of non-overlapping occurrences of string `t` in `str`.

Jack has written a function `buggyCountNonOverlappingSubstringOccurrences` that accepts `str` and `t` in its arguments. However, there are some bugs in the function logic. Fix the issues in `buggyCountNonOverlappingSubstringOccurrences` to help Jack complete his task.

Notes:

- Let `str = "ababa"` and `t = "aba"`. If we only count non-overlapping occurrences, then "aba" appears only once in "ababa". This is because, after the first occurrence is found at the start of the string, the search continues from the character after the last character of the found occurrence. Thus, after finding the first "aba", the search would continue from the second "b" in "ababa", and no further occurrences of "aba" would be found. Therefore, for `str = "ababa"` and `t = "aba"`, `buggyCountNonOverlappingSubstringOccurrences(str, t)` should return 1.

Function description
Complete the ***buggyCountNonOverlappingSubstringOccurrences*** function in the editor below. It has the following parameter(s):

Name	Type	Description
str	STRING	The given string str described in the problem.

Name	Type	Description
t	STRING	The given string t described in the problem.
Return		The function must return an *INTEGER* denoting the total number of non-overlapping occurrences of string t in str.

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
ababa aba	1	"aba" has 1 non-overlapping occurrence.
hello world world	1	world has 1 non-overlapping occurrence in "hello world".
banana na	2	na has 2 non-overlapping occurrence "nana" in the end of "banana".

```
// Fix the code here
        if (string.IsNullOrEmpty(t)) {
            return 0;
        }

        int count = 0;
        int start = 0;
        while((start = str.IndexOf(t, start)) != -1) {
            count++;
            start += 1
        }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
```

```
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggyCountNonOverlappingSubstringOccurrences(String str, String t) {
        // Fix the code here
        if (string.IsNullOrEmpty(t)) {
            return 0;
        }

        int count = 0;
        int start = 0;
        while((start = str.IndexOf(t, start)) != -1) {
            count++;
            start += t.Length;
        }

        return count;
    }

    static void Main(string[] args) {
        string str = Console.ReadLine();

        string t = Console.ReadLine();
        int result = buggyCountNonOverlappingSubstringOccurrences(str, t);

        Console.WriteLine(result);
    }
}
```

16. Fix the code to determine if a given string is a valid phone number - 2

Previous
Next

You are given a partially implemented code to determine if a given string is a valid phone number. The code reads a string and calls the function `buggyIsValidPhoneNumber` to check if the string is a valid phone number.

However, the function contains a bug that causes it to produce incorrect results. Your task is to find and fix the bug in the `buggyIsValidPhoneNumber` function.

Notes:

A valid phone number should meet the following requirements:

1. Contain exactly 10 digits, length of the input can be longer if parentheses and/or dashes included.
2. Contain only digits (0-9), optional parentheses, and optional dashes.
3. If parentheses are included, they should appear around the first three digits (e.g., "(123)456-7890").
4. If dashes are included, they should appear either between the fourth and fifth digits or after the seventh digit (e.g., "123-4567890" or "123456-7890").

Thus, if the given phone number is "1234567890", the output should be true. If the given phone number is "(123)456-7890", the output should also be true. However, if the given phone number is "123-4567-890" or "(123-456)-7890", the output should be false because the parentheses and dashes are not in the correct positions.

Function description

Complete the ***buggyIsValidPhoneNumber*** function in the editor below. It has the following parameter(s):

Name	Type	Description
s	STRING	The given phone number
Return		The function must return a *BOOLEAN* denoting the value 1 if it is a valid phone number, 0 if it is not a valid phone number.

Constraints

Input format for debugging

Sample Testcases

Input	Output	Output Description
`1234567890`	`1`	This is a valid phone number.
`12345678901`	`0`	The number is longer than 10 digits. Hence, it is not a valid number.

Input	Output	Output Description
123-456-789	0	There are two dashes in the number. Hence, it is not a valid phone number.

```
// Fix the code here
        if (s.Length == 12) {
            return Int64.TryParse(s, out _);
        } else if (s.Length == 13) {
            return s[0] == '(' && s[4] == ')' && s[8] == '-' &&
                   Int64.TryParse(s.Substring(1,3), out _) &&
                   Int64.TryParse(s.Substring(5,3), out _) &&
                   Int64.TryParse(s.Substring(9,4), out _);
        } else if (s.Length == 11 && s[3] == '-') {
            return Int64.TryParse(s.Substring(0,3), out _) &&
                   Int64.TryParse(s.Substring(4,7), out _);
        } else if (s.Length == 11 && s[6] == '-') {
            return Int64.TryParse(s.Substring(0,6), out _) &&
                   Int64.TryParse(s.Substring(7,4), out _);
        }
        return false;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static bool buggyIsValidPhoneNumber(String s) {
        // Fix the code here
        if (s.Length == 10) {
            return Int64.TryParse(s, out _);
        } else if (s.Length == 13) {
            return s[0] == '(' && s[4] == ')' && s[8] == '-' &&
```

```
                Int64.TryParse(s.Substring(1,3), out _) &&
                Int64.TryParse(s.Substring(5,3), out _) &&
                Int64.TryParse(s.Substring(9,4), out _);
        } else if (s.Length == 11 && s[3] == '-') {
            return Int64.TryParse(s.Substring(0,3), out _) &&
                Int64.TryParse(s.Substring(4,7), out _);
        } else if (s.Length == 11 && s[6] == '-') {
            return Int64.TryParse(s.Substring(0,6), out _) &&
                Int64.TryParse(s.Substring(7,4), out _);
        }
        return false;
    }

    static void Main(string[] args) {
        string s = Console.ReadLine();
        bool result = buggyIsValidPhoneNumber(s);

        Console.WriteLine(result ? 1: 0);
    }
}
```

17. Fix the code to determine if a given string is a valid credit card number - 2

Previous
Next

You are given a partially implemented code to determine if a given string is a valid credit card number. The code reads a string and calls the function `buggyIsValidCreditCard` to check if the string is a valid credit card number.

However, the function contains a bug that causes it to produce incorrect results.

A valid credit card number should:

1. Contain exactly 16 digits.
2. Start with 4, 5, or 6.
3. Contain only digits (0-9).
4. Pass the Luhn algorithm (a simple checksum formula used to validate a variety of identification numbers, such as credit card numbers).

The Luhn algorithm works as follows:

1. Double the value of every second digit from the rightmost digit.
2. If doubling a number results in a two-digit number, add the two digits to get a single-digit number.
3. Take the sum of all the single-digit numbers obtained in the previous steps.

4. If the total modulo 10 is equal to 0 (if the total ends in zero), the number is valid according to the Luhn formula; otherwise, it is not valid.

For example, if the input string is "4532015112830366", the output should be true, but if the input string is "1234567812345678", the output should be false because it does not pass the Luhn algorithm or start with 4, 5, or 6.

Your task is to find and fix the bug in the `buggyIsValidCreditCard` function.
Function description
Complete the ***buggyIsValidCreditCard*** function in the editor below. It has the following parameter(s):

Name	Type	Description
str	STRING	The credit card numbers to be validated.
Return		The function must return a *BOOLEAN* denoting the value 1 if credit card numbers are valid and 0 if it is not valid.

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
30569309025904	0	This is not a valid credit card number.
6771549495586803	0	This is not a valid credit card number.
4532015112830367	0	This is not a valid credit card number.

```
public static bool buggyIsValidCreditCard(string cardNumber) {
        // Fix the code here
        if (cardNumber.Length != 16) {
            return false;
        }
```

```
        if (cardNumber[0] != '4' && cardNumber[0] != '5' && cardNumber[0] != '6') {
            return false;
        }

        if (!cardNumber.All(char.IsDigit)) {
            return false;
        }

        // Luhn algorithm
        int sum = 0;
        for (int i = cardNumber.Length - 1; i >= 0; i--) {
            int digit = cardNumber[i] - '0';
            if (i % 2 == cardNumber.Length % 2) {
                digit *= 2;
                if (digit > 9) {
                    digit -= 9;
                }
            }
            sum += digit;
        }

        return sum % 10 != 0;
    }
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    public static bool buggyIsValidCreditCard(string cardNumber) {
        // Fix the code here
        if (cardNumber.Length != 16) {
            return false;
        }

        if (cardNumber[0] != '4' && cardNumber[0] != '5' && cardNumber[0] != '6') {
            return false;
```

```
        }

        if (!cardNumber.All(char.IsDigit)) {
            return false;
        }

        // Luhn algorithm
        int sum = 0;
        for (int i = cardNumber.Length - 1; i >= 0; i--) {
            int digit = cardNumber[i] - '0';
            if (i % 2 == cardNumber.Length % 2) {
                digit *= 2;
                if (digit > 9) {
                    digit -= 9;
                }
            }
            sum += digit;
        }

        return sum % 10 == 0;
    }

    static void Main(string[] args) {
        string str = Console.ReadLine();
        bool result = buggyIsValidCreditCard(str);

        Console.WriteLine(result ? 1: 0);
    }
}
```

18. Fix the code to find sum of the factorial of all odd numbers - 2

Previous

Next

John has an array `arr` of size `n`. He wants to find the sum of the factorial of all odd numbers in `arr` that are less than 10.

John has written a function `buggySumOfOddFactorialsLessThanTen` that accepts `n` and stores `arr` as a list of integers in its arguments. However, there are some bugs in the function logic. Fix the issues to help John complete his task.

Function description

Complete the ***buggySumOfOddFactorialsLessThanTen*** function in the editor below. It has the following parameter(s):

Name	Type	Description
n	INTEGER	The size of arr.

Name	Type	Description
arr	INTEGER ARRAY	The given array.
Return	The function must return an *INTEGER* denoting the sum of the factorial of all odd numbers in arr that are less than 10.	

Constraints
Input format for debugging
Sample Testcases

Input	Output	Output Description
4 1 2 3 4	7	Here the odd numbers less than 10 are 1 and 3. Hence, 1! + 3! = 1 + 3*2*1 = 7 is the answer.
5 5 7 9 11 13	368040	Here the odd numbers less than 10 are 5,7 and 9. Hence, 5! + 7! + 9! = 368040 is the answer.
5 2 4 6 8 10	0	Here there are no odd numbers so answer here is 0.

```
// Fix the code here
        int sum = 0;
        for(int i = 0; i < n; i++){
            if(arr[i] > 10 && arr[i] % 2 != 0)
                sum += Factorial(arr[i]);
        }
```

```
        return sum;
```

Ans:-

```
using System.CodeDom.Compiler;
using System.Collections.Generic;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Reflection;
using System.Runtime.Serialization;
using System.Text.RegularExpressions;
using System;


class Solution {
    static int buggySumOfOddFactorialsLessThanTen(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for(int i = 0; i < n; i++){
            if(arr[i] < 10 && arr[i] % 2 != 0)
                sum += Factorial(arr[i]);
        }
        return sum;
    }

    static int Factorial(int n){
        return (n == 1 || n == 0) ? 1 : Factorial(n - 1) * n;
    }

    static void Main(string[] args) {
        int n = Int32.Parse(Console.ReadLine().Trim());

        List<int> arr = new List<int>();
        for(int j=0; j<n; j++) {
            arr.Add(Int32.Parse(Console.ReadLine().Trim()));
        }
        int result = buggySumOfOddFactorialsLessThanTen(n, arr);

        Console.WriteLine(result);
    }
}
```