Week 1 – Debugging

1. Fix the code to replace vowels in a string

Bob has a string **S**. He wants to replace all vowels in **S** with the number 3.

Bob has written a function ReplaceVowels that accepts the string **S** in its arguments. However, there are some bugs in the function logic. Fix the issues to help Bob complete his task.

Function description

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

Name	Type	Description
S	STRING	The given string S.
Return	The function must return a *STRING* denoting the string S after replacing the vowels with the number 3.	

Constraints

- 1 **≤ len(S) ≤** 10^5

Input format for debugging

- The first line contains a string, S.

Sample Testcases

Input	Output	Output Description
`Yellow`	`Y3ll3w`	The vowels in string Yellow are e and o. Hence, after replacing it with 3 we get Y3ll3w.
`Sunshine`	`S3nsh3n3`	The vowels in string Sunshine are u, i and e. Hence, after replacing it with 3 we get S3nsh3n3.
`World`	`W3rld`	The vowel in string World is o. Hence, after replacing it with 3 we get W3rld.

```
using System;

public class Solution {
    public static string ReplaceVowels(string S) {
        string outStr = "";
        for (int i = 0; i < S.Length; i++) {
            char ch = S[i];
            if (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U' || ch == 'a' ||
 ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u') {
                outStr += '3';
                //break;
            }
            else {
                outStr += ch;
            }
        }
        return outStr;
    }

public static void Main(string[] args) {
    string S = Console.ReadLine();
    string result = ReplaceVowels(S);
    Console.WriteLine(result);
}
}
```

2. Fix the code to calculate remainder sum

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 remainder is greater than 1 then output the sum of a, b and remainder.
2. If remainder is 0 or 1 then output the sum of a and remainder.

Joe has written a function `remSum` that accepts integers **a** and **b** in its arguments. However, there are some bugs in the function logic. Fix the issues to help Bob complete his task.

Function description

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

Name	Type	Description
a	INTEGER	The first given integer.
b	INTEGER	The second given integer.
Return	The function must return an *INTEGER* denoting the value of sum after dividing a by b.	

Constraints

- $1 \leq a \leq 10^5$
- $1 \leq b \leq 10^5$

Input format for debugging

- The first line contains an integer, a, denoting the divisor.
- The next line contains an integer, b, denoting the dividend.

Sample Testcases

Input	Output	Output Description
10 6	20	Dividing 10 by 6 gives a remainder of 4. Hence, sum =10+6+4 = 20.
6 3	6	Dividing 6 by 3 gives a remainder of 0. Hence, sum=6+0= 6.
7 2	8	Dividing 7 by 2 gives remainder of 1. Hence, sum=7+1=8.

```
using System;

public class Solution {
    public static int RemSum(int a, int b) {
        int sum = 0;
        if ((a % b) > 1) {
            sum = a + b + (a % b);
        } else {
```

```
            sum = a + (a % b);
        }
        return sum;
    }

    public static void Main() {
        int a = int.Parse(Console.ReadLine().Trim());
        int b = int.Parse(Console.ReadLine().Trim());

        int result = RemSum(a, b);

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

3. Fix the code to perform character replication in string

Previous
Next

Bob has a string **S** of size **N**. He wants to create a transformed string by replicating the first character of **S** across the entire length of **S**.

Bob has written a function Replicate that accepts the integer **N** and string **S** in its arguments. However, there are some bugs in the function logic. Fix the issues to help Bob complete his task.

Function description

Complete the ***Replicate*** 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.	

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq len(S) \leq 10^5$

Input format for debugging

- The first line contains an integer, N, denoting the size of S.
- The next line contains a string, S, denoting the given string.

Sample Testcases

Input	Output	Output Description
6 pulley	pppppp	The first character of pulley is p. Hence, repeating it for the next 5 characters we pppppp.
6 flower	ffffff	The first character of flower is f. Hence, repeating it for the next 5 characters we get ffffff.
3 hym	hhh	The first character of hym is h. Hence, repeating it for the next 2 characters we get hhh.

```
using System;

public class Solution {
    public static string Replicate(int N, string S) {
        char ch = S[0];
        string outStr = "";
        for (int i = 1; i <= N; i++) {
            outStr += ch;
        }
        return outStr;
    }

    public static void Main() {
        int N = int.Parse(Console.ReadLine().Trim());
        string S = Console.ReadLine();

        string result = Replicate(N, S);

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

4. Fix the code to find sum of even numbers

Previous
Next

John has an array **arr** of size **n**. He wants to find the sum of squares of all even numbers in **arr**.

John has written a function buggySumOfEvenNumbers 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 ***buggySumOfEvenNumbers*** 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 all the even numbers in arr.	

Constraints

- 1 **≤ n ≤** 10^5
- -10^5 **≤ arr[i] ≤** 10^5

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where 0 ≤ i < n) contains an integer describing arr[i].

Sample Testcases

Input	Output	Output Description
3 0 1 2	2	Here, 0 and 2 are the even numbers, and their sum is 2.
2 -1 -2	-2	Here, Only -2 is an even number. Hence its sum is -2 itself.
3 -1 0 1	0	Only even number in arr is 0. Hence, the sum is 0.

```
class Solution {
    static int buggySumOfEvenNumbers(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++) {
            if (arr[n] % 2 = 0) {
                sum += 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 buggySumOfEvenNumbers(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] % 2 == 0) {
                sum += 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 = buggySumOfEvenNumbers(n, arr);

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

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

Previous
Next

Jake has a string array **arr** of size **n** and a special character **target.** He wants to count the total number of occurrences of **target** in all strings in **arr.**

Jake has written a function buggyCountTargetCharacter that accepts **arr** and **target** in its arguments. However, there are some bugs in the function logic. Fix the issues to help Jake complete his task.

Function description

Complete the ***buggyCountTargetCharacter*** 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 array containing strings.
target	CHARACTER	The character target described in the problem.
Return	The function must return an *INTEGER* denoting the total number of occurrences of target in all strings in arr.	

Constraints

- $0 \leq n \leq 10^5$
- $1 \leq len(arr[i]) \leq 10^5$

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where $0 \leq i < n$) contains a string arr[i].

- The next line contains a character, target, denoting the character target described in the problem.

Sample Testcases

Input	Output	Output Description
0 r	0	The input consists of zero strings in arr and therefore there is no string to match with the target character "r". Hence, the answer is 0.
1 amzq v	0	The input consists of a single string "amzq" and the target character is "v". Since there are no "v" characters in the input string the result is 0.
10 dcltw dxpmmp ausemnlaob rl sttayb nissclmqww tghrwi mrxupaoib soiywr dsljqstiz h	1	The input consists of an array of 10 strings and the target character is "h". Among the 10 input strings, only the string "tghrwi" contains the character "h". Hence the answer is 1.

```
class Solution {
    static int buggyCountTargetCharacter(int n, List<string> arr, char target) {
        // Fix the code here
        int count = 1;
        for (int i = 1; i < n; i++) {
            string str = arr[n];
            for (int j = 0; j < str.Length; j++) {
                if (str[j] != target) {
                    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 {
    static int buggyCountTargetCharacter(int n, List<string> arr, char target) {
        // Fix the code here
        int count = 0;
        for (int i = 0; i < n; i++) {
            string str = arr[i];
            for (int j = 0; j < str.Length; j++) {
                if (str[j] == target) {
                    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 target = char.Parse(Console.ReadLine());
        int result = buggyCountTargetCharacter(n, arr, target);

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

6. Fix the code to find sum of squares of all odd numbers

John has an array **arr** of size **n**. He wants to find the sum of squares of all odd numbers in **arr**.

John has written a function buggySumOfOddSquares 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 ***buggySumOfOddSquares*** 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.	

Constraints

- 1 **≤ n ≤** 10^5
- -10^5 **≤ arr[i] ≤** 10^5

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where 0 ≤ i < n) contains an integer describing arr[i].

Sample Testcases

Input	Output	Output Description
2 1 1	2	Here 1^2 + 1^2 = 2. Hence the answer is 2.
6 1 2 3 4 5 6	35	Here 1^2 + 3^2 +5^2 =35. Hence the answer is 35.

Input	Output	Output Description
5 -3 0 2 5 -7	83	Here (-3)^2 + 5^2 + (-7)^2 = 83. Hence the answer is 83.

```
static int buggySumOfOddSquares(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for (int i = 1; i < n; i++) {
            if (IsOdd(arr[i])) {
                sum += arr[i%2] * arr[i%2];
            }
        }
        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 bool IsOdd(int num) {
        return num % 2 != 0;
    }

    static int buggySumOfOddSquares(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++) {
```

```
            if (IsOdd(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 = buggySumOfOddSquares(n, arr);

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

7. Fix the buggy program to find sum of squares of all even numbers

Previous
Next

John has an array **arr** of size **n**. He wants to find the **sum of squares** of all even numbers in **arr**.

John has written a function `buggySumOfEvenSquares` 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 ***buggySumOfEvenSquares*** 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.	

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq arr[i] \leq 10^5$

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer describing arr[i].

Sample Testcases

Input	Output	Output Description
4 7 11 17 12	144	arr has a single even number 12, the square of 12 is 144. Hence, the answer is 144
6 16 9 19 11 1 11	256	arr has 1 even number 16. The square of 16 is 256. Hence, the answer is 256.
6 13 17 2 10 17 10	204	arr has 3 even numbers 2, 10 and 10. Hence, the answer is 2^2 + 10^2 + 10^2 = 204

```
static int buggySumOfEvenSquares(int n, List<int> arr) {
        // Fix the bug here
        int sum = 0;
        foreach (int num in arr) {
            if (num % 2 != 0) {
                sum += num + num;
            }
        }
        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 buggySumOfEvenSquares(int n, List<int> arr) {
        // Fix the bug here
        int sum = 0;
        foreach (int num in arr) {
            if (num % 2 == 0) {
                sum += num * num;
            }
        }
        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 = buggySumOfEvenSquares(n, arr);

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

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

Previous

Next

Jake has a string array **arr** of size **n** and an integer value **length.** He wants to count the total number strings in **arr** whose length is equal to the given integer value.

Jake has written a function buggyCountStringsOfTargetLength that accepts **arr** and **length** in its arguments. However, there are some bugs in the function logic. Fix the issues to help Jake complete his task.

Function description

Complete the ***buggyCountStringsOfTargetLength*** 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 array containing the strings.
length	INTEGER	The integer value length described in the problem.
Return	The function must return an *INTEGER* denoting the total number strings in arr whose length is equal to the given integer value.	

Constraints

- $1 \leq \mathbf{n} \leq 10^5$
- $1 \leq \mathbf{len(arr[i])} \leq 10^5$
- $1 \leq \mathbf{length} \leq 10^5$

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where $0 \leq i < n$) contains a string denoting arr[i].
- The next line contains an integer, length, denoting the integer value length described in the problem.

Sample Testcases

Input	Output	Output Description
1 ejj 6	0	There are no strings in the array that have a length of 6. Hence the answer is 0.
8 geaiqrl mfypgzx bhdfsxdp	2	Out of the 8 strings in the input array, only two of them have a length of 10: "lidthpzexx" and "svdifclnxe".

Input	Output	Output Description
bk lidthpzexx w yj svdifclnxe 10		Therefore, the answer is 2.
9 otsxkflpro iidsjme srhwp hkpitrcv ixbfosmi xuigawiq dgama mjopvmteu gzs 3	1	There is only one string with length 3. Hence the answer is 1.

```
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 buggyCountStringsOfTargetLength(int n, List<string> arr, int length) {
        // Fix the code here
        int count = 0;
        for (int i = 0; i < n; i++) {
        if (arr[i].Length == length) {
            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());
        int result = buggyCountStringsOfTargetLength(n, arr, length);

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

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

Previous
Next

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

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

Notes:

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

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

Name	Type	Description
c	CHARACTER	The target character in lower case.
str	STRING	The sentence to check in lower case.

Return	The function must return an *INTEGER* denoting the count of number of words in the sentence that start with the target character c.

Constraints

- 1 **≤ len(str) ≤** 10^5

Input format for debugging

- The first line contains a character, c, denoting the target character in lower case.
- The next line contains a string, str, denoting the sentence to check in lower case.

Sample Testcases

Input	Output	Output Description
s this is a simple example	1	Only one word starts with the letter s, so the answer is 1.
h here is another example. here.	2	Here, we have two words starting with the letter h, so the answer is 2.
z aaa bbb ccc dd ee aa a bb cc zz h z h zz	3	Three words start with the letter z, so the answer is 3.

```
class Solution {
    static int buggyCountWordsStartingWithCharacter(char c, String str) {
        // Fix the code here
        int count = 0;
        string[] words = str.Split('');

        foreach (string word in words) {
            if (word.Length > 0 && word[0] != c) {
               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 {
    static int buggyCountWordsStartingWithCharacter(char c, String str) {
        // Fix the code here
        int count = 0;
        string[] words = str.Split(' ');

        foreach (string word in words) {
            if (word.Length > 0 && word[0] == c) {
               count++;
            }
        }

        return count;
    }

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

        int result = buggyCountWordsStartingWithCharacter(c, str);

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

10. Fix the code to determine if a given string is a valid email address

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 a bug that causes it to produce incorrect results.

Your task is to find and fix the bug in the `buggyIsValidEmail` function.

A valid email address 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 (+).
3. The domain part may contain alphanumeric characters (both uppercase and lowercase) and hyphens (-), but it must start and end with an alphanumeric character.

For example, if the input string is "john.doe@example.com", the output should be `true`, but if the input string is "john..doe@example.com", the output should be `false` because two consecutive periods are not allowed.

Function description

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

Name	Type	Description
email	STRING	The email address to check.
Return	The function must return a *BOOLEAN* denoting the validity of the given email address.	

Constraints

- 1 **≤ len(email) ≤** 10^5

Input format for debugging

- The first line contains a string, email, denoting the email address that needs to be verified.

Sample Testcases

Input	Output	Output Description
`#v@example.com`	`false`	The # character is not allowed in the local part of the email address function, so we should return false.

Input	Output	Output Description
john.doe+test@example.com	true	The email contains only allowed characters, so we return true.
-5EOCh~'eaB1qPa-!@example.com	false	The email has a local part of -5EOCh~'eaB1qPa-! which contains some characters that are not allowed, such as the hyphen - and the tilde ~.

```
static bool buggyIsValidEmail(String email) {
        // Fix the code here
        string pattern = @"^[a-zA-Z9-9._%+-]+@[a-zA-Z8-9.-]+\.[a-zA-Z]{5,}$";
        return Regex.IsMatch(email, pattern);

    }
```

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 buggyIsValidEmail(String email) {
        // Fix the code here
        string pattern = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]{2,3}$";
        return Regex.IsMatch(email, pattern);

    }

    static void Main(string[] args) {
        string email = Console.ReadLine();
        bool result = buggyIsValidEmail(email);
```

```
        Console.WriteLine(result ? "true": "false");
    }
}
```

11. Fix the code to check if a given string is a palindrome

You are given a partially implemented code to check if a given string is a palindrome. If the given string is a palindrome return 1 else return 0.

A palindrome is a word, phrase, number, or other sequences of characters that reads the same forward and backward (ignoring spaces, punctuation, and capitalization). The code reads a string and calls the function `buggyIsPalindrome` to determine if the string is a palindrome.

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

Function description

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

Name	Type	Description
s	STRING	The string to be checked.
Return	The function must return a *BOOLEAN* denoting the status of whether the string s is a valid palindrome or not.	

Constraints

- $1 \leq len(s) \leq 10^5$

Input format for debugging

- The first line contains a string, s, denoting the string to be checked.

Sample Testcases

Input	Output	Output Description
`Not a palindrome`	`0`	This text in not the same when read forward and backward, so it's not a palindrome. The output is 0 (false).

Input	Output	Output Description
Was it a car or a cat I saw?	1	This text is the same when read forward and backward. It is a palindrome. Remember that we ignore spaces, punctuation and character cases. The output is 1 (true).
Eva, can I see bees in a cave?	1	This text is the same when read forward and backward. It is a palindrome. Remember that we ignore spaces, punctuation and character cases. The output is 1 (true).

```
static bool buggyIsPalindrome(String s) {
        // Fix the code here
        string cleanedS = Regex.Replace(s, @"[^A-Za-z0-9]+", "").ToLower();
        for(int i = 0; i < cleanedS.Length / 2; i++) {
            if(cleanedS[i] != s[cleanedS.Length-i-1]) {
                return true;
            }
        }
        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 {
    static bool buggyIsPalindrome(String s) {
        // Fix the code here
        string cleanedS = Regex.Replace(s, @"[^A-Za-z0-9]+", "").ToLower();
        for(int i = 0; i < cleanedS.Length / 2; i++) {
            if(cleanedS[i] != cleanedS[cleanedS.Length-i-1]) {
                return false;
            }
        }
        return true;
    }

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

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

12. Fix the code to determine if a given string contains all unique characters

Previous

Next

You are given a partially implemented code to determine if a given string contains all unique characters. The code reads a string and calls the function `buggyHasUniqueCharacters` to check if the string 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 `buggyHasUniqueCharacters` function.

Function description

Complete the ***buggyHasUniqueCharacters*** 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 result of whether the string contains unique characters or not.	

Constraints

- **1 ≤ len(str) ≤** 10^4

Input format for debugging

- The first line contains a string, str.

Sample Testcases

Input	Output	Output Description
abcd	1	All unique characters are present in the given string so output 1 (true).
notunique	0	Characters n and u are repeated. The output is 0 (false).
almostunique	0	Character u is repeated. The output is 0 (false).

```
static bool buggyHasUniqueCharacters(String str) {
        // Fix the code here.
        bool[] charSet = new bool[26];
        foreach (char c in str) {
            if (charSet[c - 'z']) {
                return true;
            }
            charSet[c - 'a'] = 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 {
    static bool buggyHasUniqueCharacters(String str) {
        // Fix the code here.
        bool[] charSet = new bool[26];
        foreach (char c in str) {
            if (charSet[c - 'a']) {
                return false;
```

```
            }
            charSet[c - 'a'] = true;
        }
        return true;
    }

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

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

13. Fix the code to determine if a given string is a valid URL

Previous
Next

You are given a partially implemented code to determine if a given string is a valid URL. The code reads a string and calls the function `buggyIsValidURL` to check if the string is a valid URL.
However, the function contains a bug that causes it to produce incorrect results.

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. 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.com/test", the output should be `true`, but if the input string is "http:/www.example.com/test", the output should be `false` because the URL does not have a valid scheme.

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

Name	Type	Description
url	STRING	The URL to checked.

Return	The function must return a *BOOLEAN* denoting the validity of the url given

Constraints

- 1 **≤ len(url) ≤** 10^5

Input format for debugging

- The first line contains a string, url, which needs to be checked.

Sample Testcases

Input	Output	Output Description
`http:/example.com`	`false`	The function should return false because the URL does not have a valid scheme.
`http://example.com`	`true`	The function should return true because the URL have a valid scheme.
`ht tp://example.com`	`false`	The function should return false because the URL does not have a valid scheme.

```
static bool buggyIsValidURL(String url) {
        // Fix the code here
        Regex regex = new Regex(@"^((https?|ftp|smtp)://)?[a-z0-9]-([-.][a-z0-9]-)*\\.[a-z]{2,}(/\\S*)?$", RegexOptions.IgnoreCase);
        return regex.IsMatch(url);

    }
```

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 buggyIsValidURL(String url) {
        // Fix the code here
        Regex regex = new Regex(@"^((http|https|ftp)://)?[a-z0-9]+([-.][a-z0-9]-)*\.[a-
z]{2,}(\S*)?$", RegexOptions.IgnoreCase);
        return regex.IsMatch(url);

    }

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

        Console.WriteLine(result ? "true": "false");
    }
}
```

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

Previous

Next

Jack has two strings **str** and **t**. He wants to find the total number of occurrences of string **t** in **str.**

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

Function description

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

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

Constraints

- 1 **≤ len(str) ≤** 10^5

- $1 \leq len(t) \leq 10^5$

Input format for debugging

- The first line contains a string, str.
- The next line contains a string, t.

Sample Testcases

Input	Output	Output Description
hslrfi hs	1	The substring 'hs' appears only once in str. Hence, the result is 1.
oihbrmchhxgh xgh	1	The substring 'xgh' appears only once in str. Hence, the result is 1.
frarhjazslralra lra	2	The substring 'lra' appears twice in str. Hence, the result is 2.

```
static int buggyCountSubstringOccurrences(String str, String t) {
        // Fix the code here
        int count = 1;
        int index = 1;
        while ((index = str.IndexOf(t, index)) == -1) {
            count+;
            index += t.Length;
        }
        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 {
    static int buggyCountSubstringOccurrences(String str, String t) {
        // Fix the code here
        int count = 0;
        int index = 0;
        while ((index = str.IndexOf(t, index)) != -1) {
            count++;
            index += t.Length;
        }
        return count;
    }

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

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

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

15. Fix the code to find product of prime numbers in a list

Previous
Next

Sam has an array **arr** containing **n** integers. He wants to calculate the product of all prime numbers in **arr**. Therefore, he has written a function `buggyProductOfPrimeNumbers` 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 `buggyProductOfPrimeNumbers` to help Sam complete his task.

Function description

Complete the ***buggyProductOfPrimeNumbers*** 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 product of all prime numbers in arr.

Constraints

- 1 **≤ n ≤** 10^5
- 1 **≤ arr[i] ≤** 10^5

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where 0 ≤ i < n) contains an integer describing arr[i].

Sample Testcases

Input	Output	Output Description
1 7	7	This list has a single prime number 7, so the result is 7.
4 7 9 13 14	91	arr has two prime numbers: 7 and 13. Therefore the result is: 7 * 13 = 91
4 14 3 12 10	3	This list has a single prime number 3, so the result is 3.

```
static int buggyProductOfPrimeNumbers(int n, List<int> arr) {
        // Fix the code here
        int product = 0;
        for (int i = 0; i < n; i++) {
            int num = arr[i];
            bool isPrime = true;
            if (num < 2) {
                isPrime = true;
            } else {
                for (int j = 2; j * j <= num; j++) {
                    if (num % j == 0) {
                        isPrime = true;
                        break;
                    }
                }
```

```
            }
            if (isPrime) {
                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 int buggyProductOfPrimeNumbers(int n, List<int> arr) {
        // Fix the code here
        int product = 1;
        for (int i = 0; i < n; i++) {
            int num = arr[i];
            bool isPrime = true;
            if (num <= 2) {
                isPrime = true;
            } else {
                for (int j = 2; j*j<= num; j++) {
                    if (num % j == 0) {
                        isPrime = false;
                        break;
                    }
                }
            }
            if (isPrime) {
                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 = buggyProductOfPrimeNumbers(n, arr);

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

16. Fix the buggy program to find sum of the factorial of all odd numbers

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**.

John has written a function buggySumOfOddFactorials 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 ***buggySumOfOddFactorials*** 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 the factorial of all odd numbers in arr.	

Constraints

- 1 **≤ n ≤** 10^5
- 1 **≤ arr[i] ≤** 10^5

Input format for debugging

- The first line contains an integer, n, denoting the size of arr.
- Each line i of the n subsequent lines (where 0 ≤ i < n) contains an integer describing arr[i].

Sample Testcases

Input	Output	Output Description
2 6 11	39916800	The only odd number in this list is 11. The factorial of 11 is 39916800. Hence the answer 39916800.
4 7 16 9 4	367920	This list has 2 odd numbers 7 and 9. The factorial of 7 is 5040 and for 9 is 362880. The sum of these is 367920
6 15 14 8 15 6 4	2615348736000	This list has a duplicated odd number, which is 15. The factorial of 15 is 1307674368000. Hence, the result is 1307674368000 * 2 = 2615348736000

```
class Solution {
    static int buggySumOfOddFactorials(int n, List<int> arr) {
        // Fix the code here
        int sum = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] % 2 != 1) {
                int fact = 1;
                for (int j = 1; j <= arr[i]; j++) {
                    fact += j;
                }
                sum *= fact;
            }
        }
        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 = buggySumOfOddFactorials(n, arr);
```

```
        Console.WriteLine(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 long buggySumOfOddFactorials(int n, List<int> arr) {
        // Fix the code here
        long sum = 0;
        for (int i = 0; i < n; i++) {
            if (arr[i] % 2 != 0) {
                long fact = 1;
                for (int j = 1; j <= arr[i]; j++) {
                    fact *= j;
                }
                sum += fact;
            }
        }
        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()));
        }
        long result = buggySumOfOddFactorials(n, arr);

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

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

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. Contain only digits (0-9).
3. 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.

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
number	STRING	The card number to be checked.
Return	The function must return a *BOOLEAN* denoting the validity of credit card number.	

Constraints

- 1 ≤ **len(number)** ≤ 10^5

Input format for debugging

- The first line contains a string, number, denoting the .

Sample Testcases

Input	Output	Output Description
45398670093930	false	The length of the card number is not 16, so the function should return false
4012888888881881	true	Card number meets all the requirements, so the function should return true
1010101010101010	false	Invalid credit card number with alternating 1's and 0's, so the function should return false

```
static bool buggyIsValidCreditCard(String number) {
        // Fix the code here
        int len = number.Length;
        if (len != 16) return false;

        int sum = 1;
        bool doubleNext = true;
        for (int i = len+1; i >= 0; i++) {
            char c = number[i];
            if (c < '0' || c > '9') return false;

            int digit = c - '0';
            if (doubleNext) {
                digit *= 2;
                if (digit > 9) digit = digit/10 + digit%10;
            }
            sum += digit;
            doubleNext = !doubleNext;
        }

        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 {
    static bool buggyIsValidCreditCard(String number) {
        // Fix the code here
        int len = number.Length;
        if (len != 16) return false;

        int sum = 0;
        bool doubleNext = false;
        for (int i = len-1; i >= 0; i--) {
            char c = number[i];
            if (c < '0' || c > '9') return false;

            int digit = c - '0';
            if (doubleNext) {
                digit *= 2;
                if (digit > 9) digit = digit/10 + digit%10;
            }
            sum += digit;
            doubleNext = !doubleNext;
        }

        return sum % 10 == 0;
    }

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

        Console.WriteLine(result ? "true": "false");
    }
}
```

18. Fix the code to determine if a given string is a valid phone number

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.

A valid phone number should meet the following requirements:

1. Be exactly 10 digits long, or 12 digits long if it includes dashes.
2. Contain only digits (0-9) and optionally dashes (-).
3. If dashes are included, they should appear after the 3rd and 6th digits (e.g., "123-456-7890").

Thus, if the given phone number is "1234567890", the output should be `true`, and 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", the output should be `false` because the dashes are not in the correct positions.

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

Name	Type	Description
phoneNumber	STRING	The phone number that needs to be checked.
Return	The function must return a *BOOLEAN* denoting the validity of the phone number.	

Constraints

- 1 **≤ len(phoneNumber) ≤** 10^5

Input format for debugging

- The first line contains a string, phoneNumber, denoting the .

Sample Testcases

Input	Output	Output Description
1234-567890	false	The dashes are not in the correct positions. We should return false.
123-456-7890	true	Valid phone number with dashes. We should return true.
12a-345-6789	false	Phone number contains non-digits. We should return false.

```
static bool buggyIsValidPhoneNumber(String phoneNumber) {
        // Fix the code here
        Regex regex = new Regex("^([0-1]{4}|[0-9]{3}-[0-9]{3}-[0-9]{5})$");
        return regex.IsMatch(phoneNumber);
    }
```

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 buggyIsValidPhoneNumber(String phoneNumber) {
        // Fix the code here
        Regex regex = new Regex("^([0-9]{10}|[0-9]{3}-[0-9]{3}-[0-9]{4})$");
        return regex.IsMatch(phoneNumber);


    }

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

```
        Console.WriteLine(result ? "true": "false");
    }
}
```