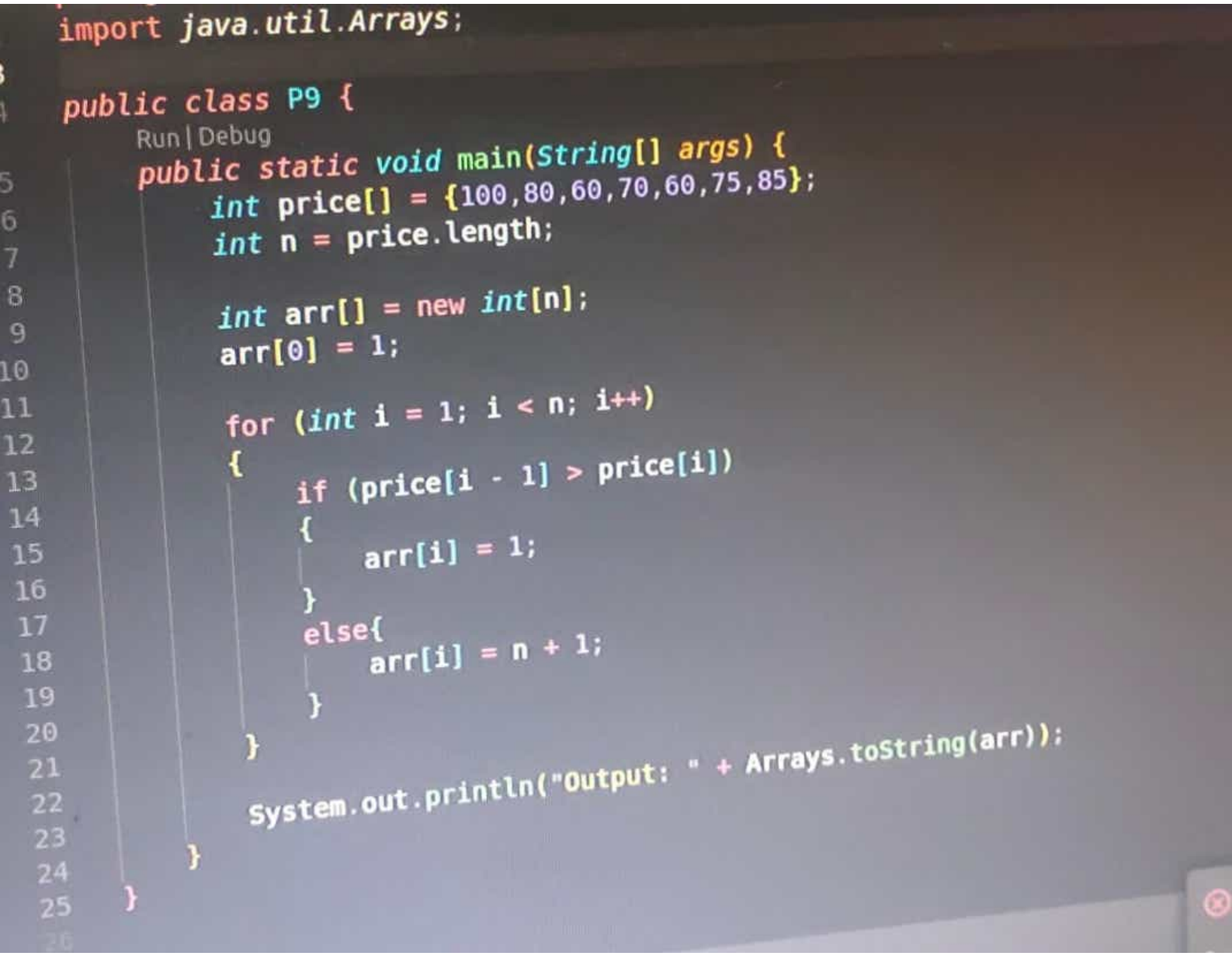


```java
import java.util.Arrays;

public class P9 {
    Run | Debug
    public static void main(String[] args) {
        int price[] = {100,80,60,70,60,75,85};
        int n = price.length;

        int arr[] = new int[n];
        arr[0] = 1;

        for (int i = 1; i < n; i++)
        {
            if (price[i - 1] > price[i])
            {
                arr[i] = 1;
            }
            else{
                arr[i] = n + 1;
            }
        }

        System.out.println("Output: " + Arrays.toString(arr));
    }
}
```

Cisco

1)Missing number in array

Given an array of size N-1 such that it only contains distinct integers in the range of 1 to N. Find the missing element.
Example 1:
Input:
N = 5
A[] = {1,2,3,5}
Output: 4

Example 2:
Input:
N = 10
A[] = {6,1,2,8,3,4,7,10,5}
Output: 9

L&T

1.

Fanny's Occurences

Fanny is given a string along with the string which contains a single character x. She has to remove the character x from the given string. Help her write a function to remove all occurrences of the x character from the given string

.

Input Specification:

input1: input string s

input2: String containing any character x

Output Specification:

String without the occurrence of character x

Example 1:

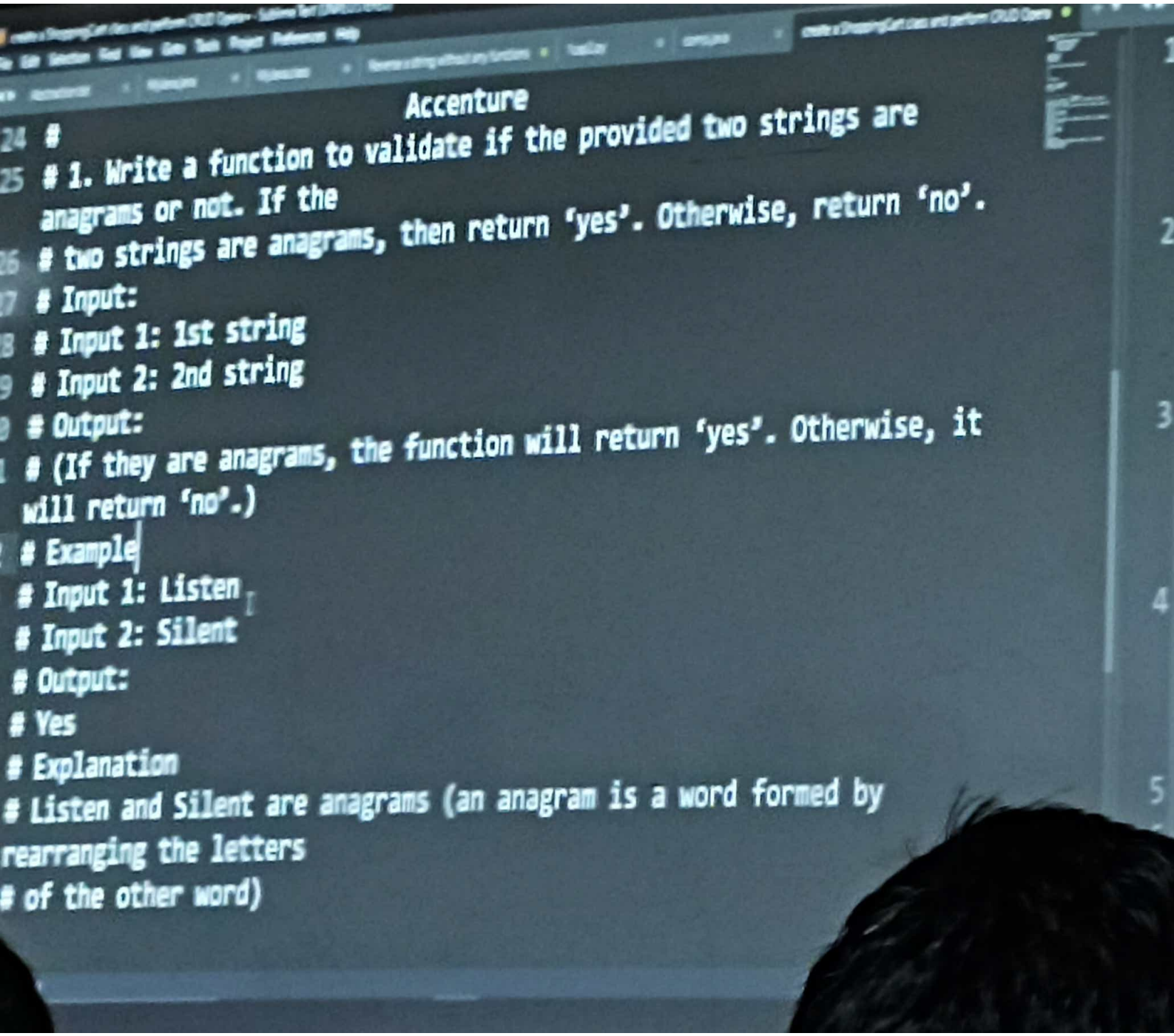
input1: welcome to metti

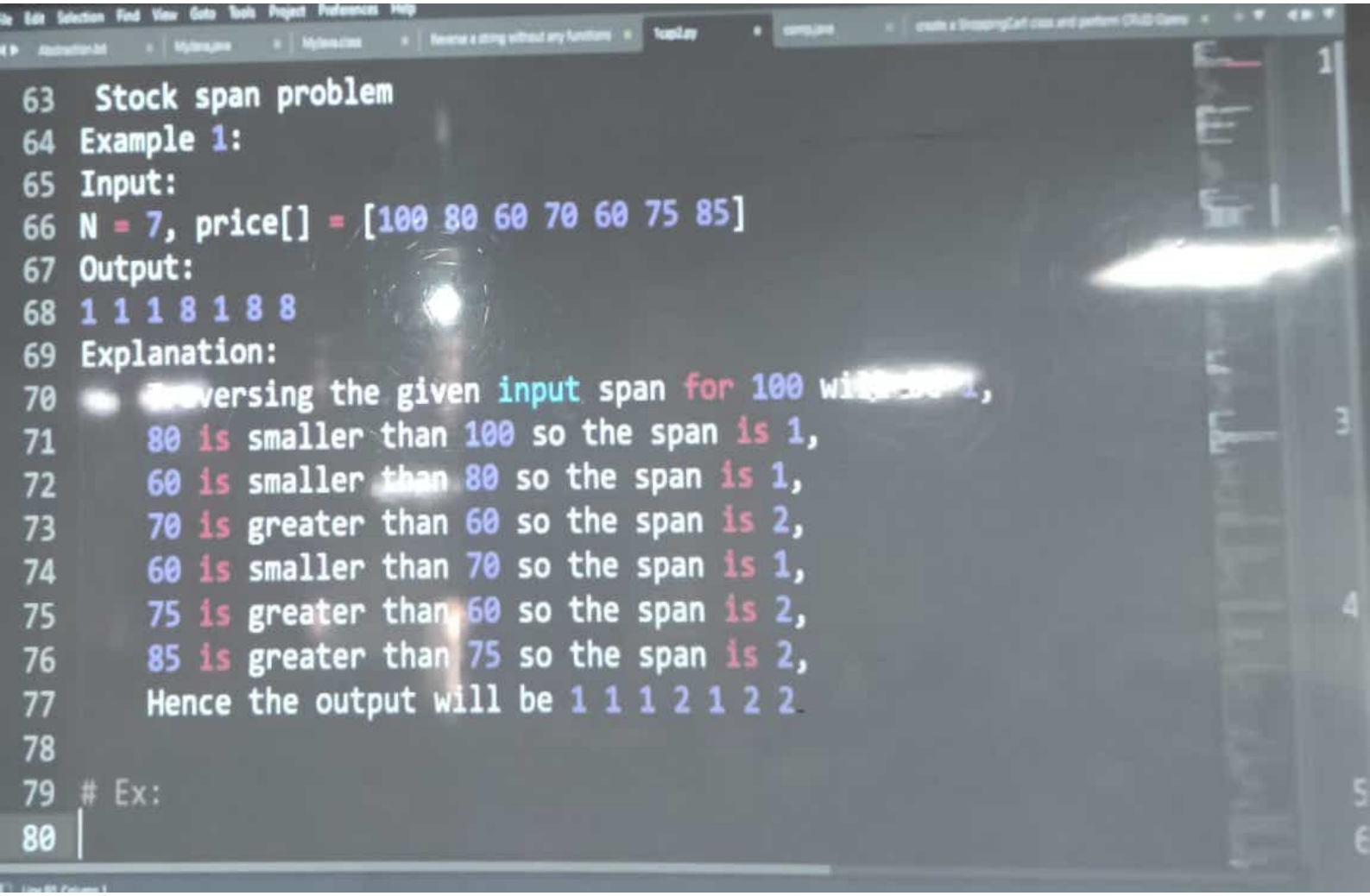
input2: i

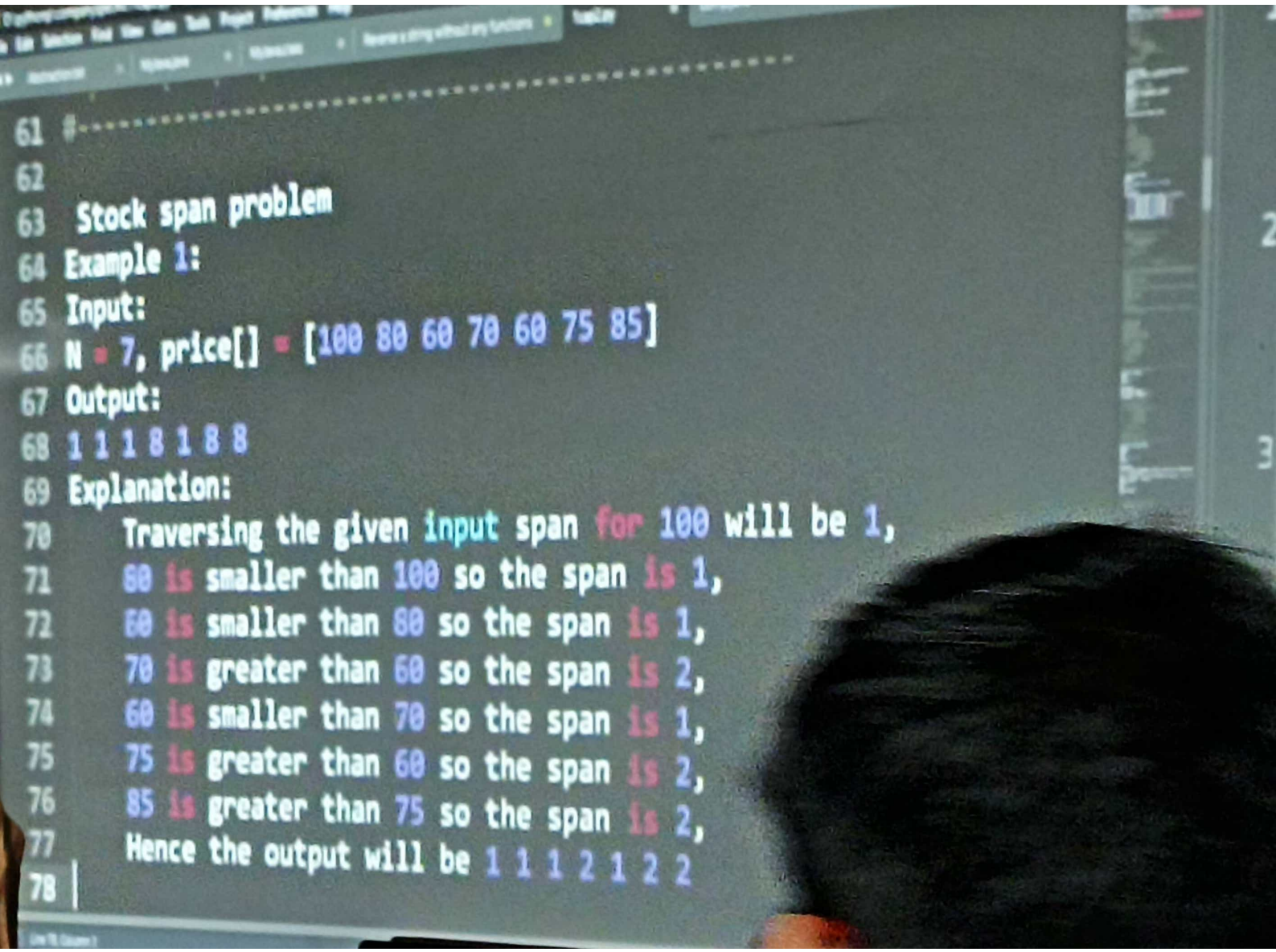
Output: wecome to mett

Explanation: As I is the character that is required to be removed, therefore all the occurrences of I are removed, keeping all other characters

```
24 #                                    Accenture
25 # 1. Write a function to validate if the provided two strings are
   anagrams or not. If the
26 # two strings are anagrams, then return 'yes'. Otherwise, return 'no'.
27 # Input:
28 # Input 1: 1st string
29 # Input 2: 2nd string
30 # Output:
31 # (If they are anagrams, the function will return 'yes'. Otherwise, it
   will return 'no'.)
32 # Example
33 # Input 1: Listen
34 # Input 2: Silent
35 # Output:
36 # Yes
37 # Explanation
38 # Listen and Silent are anagrams (an anagram is a word formed by
   rearranging the letters
39 # of the other word)
40
```


Accenture
1. Write a function to validate if the provided two strings are anagrams or not. If the
two strings are anagrams, then return 'yes'. Otherwise, return 'no'.
Input:
Input 1: 1st string
Input 2: 2nd string
Output:
(If they are anagrams, the function will return 'yes'. Otherwise, it will return 'no'.)
Example
Input 1: Listen
Input 2: Silent
Output:
Yes
Explanation
Listen and Silent are anagrams (an anagram is a word formed by rearranging the letters
of the other word)


Stock span problem
Example 1:
Input:
N = 7, price[] = [100 80 60 70 60 75 85]
Output:
1 1 1 8 1 8 8
Explanation:
80 is smaller than 100 so the span is 1,
60 is smaller than 80 so the span is 1,
70 is greater than 60 so the span is 2,
60 is smaller than 70 so the span is 1,
75 is greater than 60 so the span is 2,
85 is greater than 75 so the span is 2,
Hence the output will be 1 1 1 2 1 2 2
Ex:


#----------
Stock span problem
Example 1:
Input:
N = 7, price[] = [100 80 60 70 60 75 85]
Output:
1 1 1 2 1 2 2
Explanation:
Traversing the given input span for 100 will be 1,
80 is smaller than 100 so the span is 1,
60 is smaller than 80 so the span is 1,
70 is greater than 60 so the span is 2,
60 is smaller than 70 so the span is 1,
75 is greater than 60 so the span is 2,
85 is greater than 75 so the span is 2,
Hence the output will be 1 1 1 2 1 2 2

```
class Test
{
  public static void main(String[] args)
  {
    //conversion of String to StringBuffer
        String str1="Ashish";
        StringBuffer sb1 = new StringBuffer(str1);
        System.out.println(sb1);
    //conversion of StringBuffer to String
        StringBuffer sb2 = new StringBuffer("Prashant");
        String str2 = sb2.toString();
        System.out.println(str2);
  }
}
```

String is immutable for several reasons

- Security
- Synchronization and concurrency
- Caching
- Class loading

Difference between StringBuffer and StringBuilder

StringBuffer	StringBuilder
It is Synchronized	It is not synchronized
Object must be used in single thread programming model application	Object must be used in multithreaded programming model application

When should we go for *String, StringBuffer and StringBuilder?*

- If we don't want to store string modification in same memory, must use String
- If we want to store modification in same memory, must use StringBuffer or StringBuilder

In java, objects of String are immutable which means a constant and cannot be changed in the same memory after they are created. Hence String is defined as an immutable sequence of characters.

Immutability vs. Mutability

String is **immutable** class it means once we are creating String objects it is not possible to perform modifications on existing object.

StringBuffer & StringBuilder are **mutable** classes it means once we are creating StringBuffer objects, it is possible to perform modification on that existing object

String is a sequence of characters placed in double quotes (" ").
Performing different operations on strings is called **string handling**.

In String manipulations we are going to learn following classes

- Java.lang.String
- Java.lang.StringBuffer
- Java.lang.StringBuilder
- Java.util.StringTokenizer

Java.lang.String

S[redacted]g is used to represent group of characters or character array e[redacted]sed with in the double quotes

There are two ways to create string in Java:

String literal

String s = "Java Programming";

Using new keyword

String s = new String ("Java Programming");

```
File  Edit  Format  View  Help
class Test
{
  public static void main(String[] args)
  {
        String str="help4code";
        System.out.println(str);

        String str1=new String("help4code");
        System.out.println(str1);

        char[] ch={'h','e','l','p','4','c','o','d','e'};
        String str3=new String(ch);
        System.out.println(str3);

        char[] ch1={'a','s','h','i','h','e','l','p','4','c','o','d','e'};
        String str4=new String(ch1,2,8);
        System.out.println(str4);

        byte[] b={65,66,67,68,69,70};
```

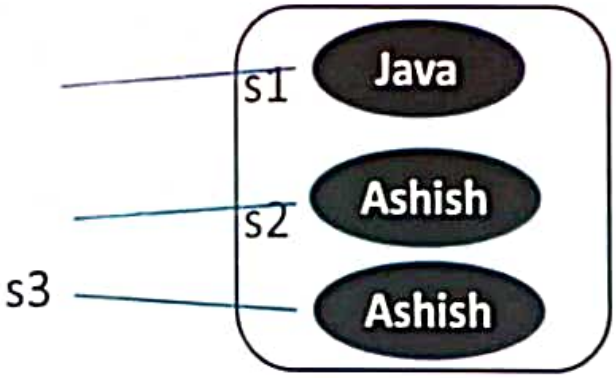
```
class Test
{
  public static void main(String[] args)
  {
        String str="help4code";
        System.out.println(str);

        String str1=new String("help4code");
        System.out.println(str1);

        char[] ch={'h','e','l','p','4','c','o','d','e'};
        String str3=new String(ch);
        System.out.println(str3);

        char[] ch1={'a','s','h','i','h','e','l','p','4','c','o','d','e'};
        String str4=new String(ch1,2,8);
        System.out.println(str4);

        byte[] b={65,66,67,68,69,70};
```

```
        String str3=new String(ch);
        System.out.println(str3);

        char[] ch1={'a','s','h','i','h','e','l','p','4','c','o','d','e'};
        String str4=new String(ch1,2,8);
        System.out.println(str4);

        byte[] b={65,66,67,68,69,70};
        String str5=new String(b);

        System.out.println(str5);
        byte[] b1={65,66,67,68,69,70};
        String str6=new String(b1,2,4);

        System.out.println(str6);
    }
}
```

Creating a string with using new operator

Whenever we are creating String object by using new operator the object created in heap area.

```
String s1 = new String("Java");
String s2 = new String( "Ashish");
String s3 = new String( "Ashish");
```



- When we create object in Heap area instead of checking previous objects it directly creates new objects
- Heap memory allows duplicate objects

1.java - Notepad

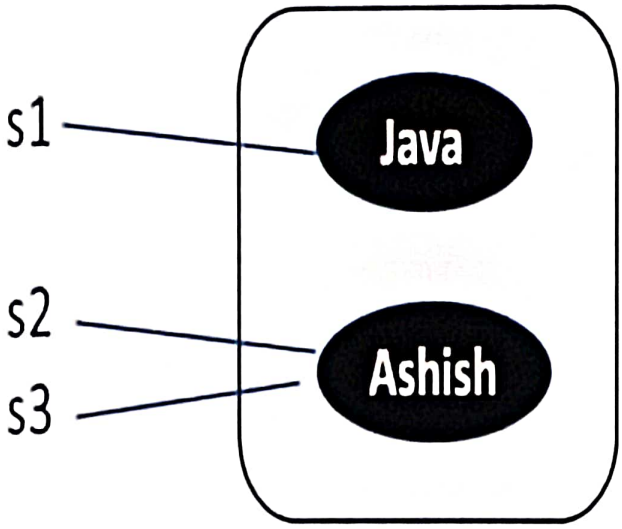
File Edit Format View Help

```
        System.out.println(str1);

        char[] ch={'h','e','l','p','4','c','o','d','e'};
        String str3=new String(ch);
        System.out.println(str3);

        char[] ch1={'a','s','h','i','h  e','l','p','4','c','o','d','e'};
        String str4=new String(ch1,2,8);
        System.out.println(str4);

        byte[] b={65,66,67,68,69,70};
        String str5=new String(b);

        System.out.println(str5);
        byte[] b1={65,66,67,68,69,70};
        String str6=new String(b1,2,4);

        System.out.println(str6);
}
```

Creating a string without using new operator

When we create String object without using **new** operator the objects are created in String constant pool area.

```
String s1 = "Java";
String s2 = "Ashish";
String s3 = "Ashish";
```



- If previous object is available with the same content then it won't create new object, that reference variable will point to existing object.
- If previous objects are not available then JVM will create new object.

```
class Test
{
  public static void main(String[] args)
  {
        Test t1 = new Test();
        Test t2 = new Test();
        System.out.println(t1==t2);
        String str1="Ashish";
        String str2="Ashish";
        System.out.println(str1==str2);
        String s1 = new String("help4code");
        String s2 = new String("help4code");
        System.out.println(s1==s2);
  }
}
```

```
class Test
{
 public static void main(String[] args)
 {
        Test t1 = new Test();
        Test t2 = new Test();
        System.out.println(t1==t2);
        String str1="Ashish";
        String str2="Ashish";
        System.out.println(str1==str2);
        String s1 = new String("help4code");
        String s2 = new String("help4code");
        System.out.println(s1==s2);
}
```

Output

```
false
true
false
```

In java, objects of String are immutable which means a constant and cannot be changed in the same memory after they are created. Hence String is defined as an immutable sequence of characters.

Immutability vs. Mutability

String is **immutable** class it means once we are creating String objects it is not possible to perform modifications on existing object.

StringBuffer & StringBuilder are **mutable** classes it means once we are creating StringBuffer objects, it is possible to perform modification on that existing object