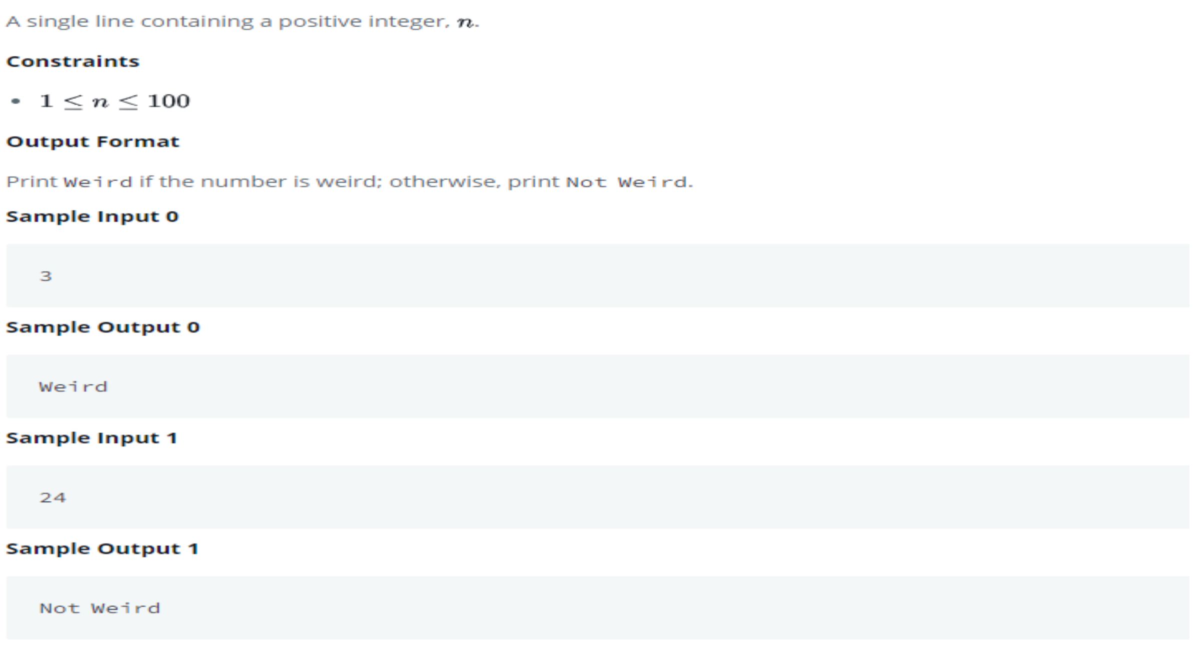
NAMA : JAMALUDIN

TYPE : EASY

---

1. **Question “Java If-Else”**



A single line containing a positive integer, n.

Constraints

- $1 \leq n \leq 100$

Output Format

Print `Weird` if the number is weird; otherwise, print `Not Weird`.

Sample Input 0

```
3
```

Sample Output 0

```
Weird
```

Sample Input 1

```
24
```

Sample Output 1

```
Not Weird
```

Answer

```
import java.io.*;

import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

public class Solution {


    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int N = scanner.nextInt();
        String ans ="";
```

```java
        if(N >= 1 && N<=100){
            if(N%2==1){
                System.out.print("Weird");
            }
        else{
            if(N>=2 && N<=5){
                System.out.print("Weird");
            }
            else if(N>20){
                System.out.print("Not Weird");
            }
            }
        }
        else{
            System.out.print("Invalid");
        }
}
}
```

2. **Question “Java 1D Array”**

Sample Input

```
5
10
20
30
40
50
```

Sample Output

```
10
20
30
40
50
```

Explanation

When we save each integer to its corresponding index in a, we get $a = [10, 20, 30, 40, 50]$. The locked code prints each array element on a new line from left to right.

<u>Answer</u>

```java
import java.util.*;

public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
              int n = scan.nextInt();
              int[] a=new int[n];
              for(int i=0;i<n;i++)
          {
               a[i]=scan.nextInt();
           }
              scan.close();

               for (int i = 0; i < a.length; i++) {
                  System.out.println(a[i]);
              }
          }
      }
```

3. Question “Java Abstract Class”

A Java abstract class is a class that can't be instantiated. That means you cannot create new instances of an abstract class. It works as a base for subclasses. You should learn about Java Inheritance before attempting this challenge.

Following is an example of abstract class:

```
abstract class Book{
    String title;
    abstract void setTitle(String s);
    String getTitle(){
        return title;
    }
}
```

If you try to create an instance of this class like the following line you will get an error:

```
Book new_novel=new Book();
```

You have to create another class that extends the abstract class. Then you can create an instance of the new class.

Notice that setTitle method is abstract too and has no body. That means you must implement the body of that method in the child class.

In the editor, we have provided the abstract Book class and a Main class. In the Main class, we created an instance of a class called MyBook. Your task is to write just the MyBook class.

Your class mustn't be public.

<u>Answer</u>

```
import java.util.*;
abstract class Book{
    String title;
    abstract void setTitle(String s);
    String getTitle(){
        return title;
    }
}

class MyBook extends Book{
     void setTitle(String s) {
        title = s;
     }
}


public class Main{

    public static void main(String []args){
        //Book new_novel=new Book(); This line prHMain.java:25: error:
Book is abstract; cannot be instantiated
        Scanner sc=new Scanner(System.in);
        String title=sc.nextLine();
        MyBook new_novel=new MyBook();
        new_novel.setTitle(title);
        System.out.println("The title is: "+new_novel.getTitle());
        sc.close();

    }
}
```

4. Question "Java List"

For this problem, we have 2 types of queries you can perform on a List:

1. Insert y at index x:

```
Insert
x y
```

2. Delete the element at index x:

```
Delete
x
```

Given a list, L, of N integers, perform Q queries on the list. Once all queries are completed, print the modified list as a single line of space-separated integers.

Answer

```
import java.io.*;
```

```java
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
       Scanner scan = new Scanner(System.in);
        int N = scan.nextInt();
        LinkedList<Integer> list = new LinkedList<>();
        for (int i = 0; i < N; i++) {
            int value = scan.nextInt();
            list.add(value);
        }

        int Q = scan.nextInt();
        for (int i = 0; i < Q; i++) {
            String action = scan.next();
            if (action.equals("Insert")) {
                int index = scan.nextInt();
                int value = scan.nextInt();
                list.add(index, value);
            } else {
                int index = scan.nextInt();
                list.remove(index);
            }
        }
        scan.close();

        for (Integer num : list) {
            System.out.print(num + " ");
        }
    }
}
```

5. Question “Java Method Overriding”

When a subclass inherits from a superclass, it also inherits its methods; however, it can also override the superclass methods (as well as declare and implement new ones). Consider the following Sports class:

```
class Sports{
    String getName(){
        return "Generic Sports";
    }
    void getNumberOfTeamMembers(){
        System.out.println( "Each team has n players in " + getName() );
    }
}
```

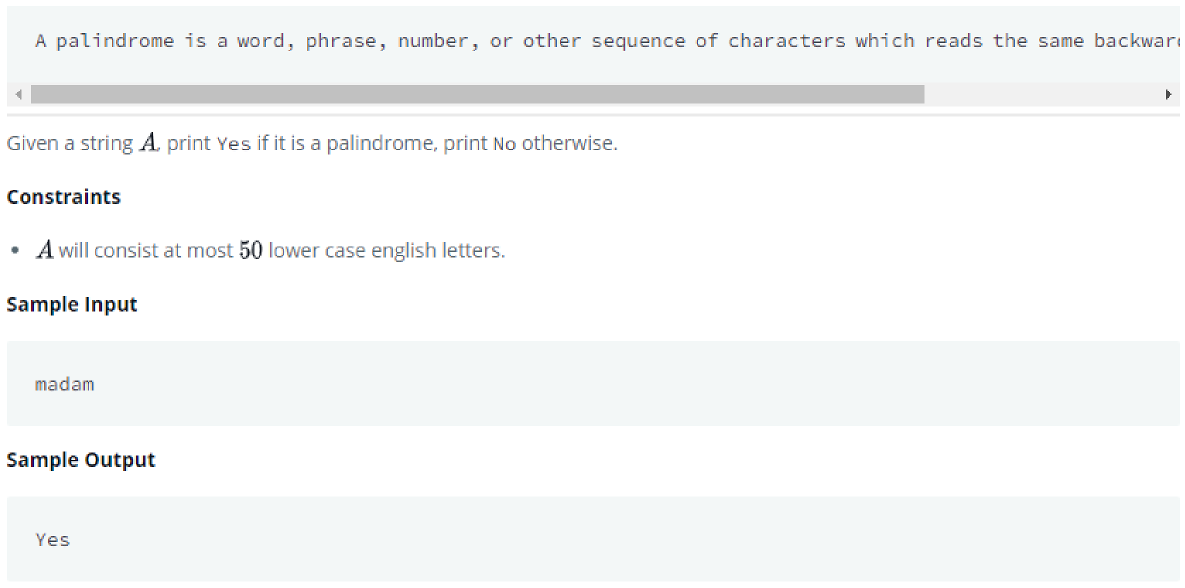
Next, we create a Soccer class that inherits from the Sports class. We can override the getName method and return a different, subclass-specific string:

```
class Soccer extends Sports{
    @Override
    String getName(){
        return "Soccer Class";
    }
}
```

Answer

```
import java.util.*;
class Sports{

    String getName(){
        return "Generic Sports";
    }

    void getNumberOfTeamMembers(){
        System.out.println( "Each team has n players in " + getName() )
;
    }
}

class Soccer extends Sports{
    @Override
    String getName(){
        return "Soccer Class";
    }
 @Override
    void getNumberOfTeamMembers(){
        System.out.println( "Each team has 11 players in " + getName()
);
    }
}

public class Solution{
    public static void main(String []args){
        Sports c1 = new Sports();
        Soccer c2 = new Soccer();
        System.out.println(c1.getName());
        c1.getNumberOfTeamMembers();
        System.out.println(c2.getName());
        c2.getNumberOfTeamMembers();
    }
}
```

6. Question "Java String Reverse"



```
A palindrome is a word, phrase, number, or other sequence of characters which reads the same backwar
```

Given a string A, print `Yes` if it is a palindrome, print `No` otherwise.

Constraints

- A will consist at most 50 lower case english letters.

Sample Input

```
madam
```

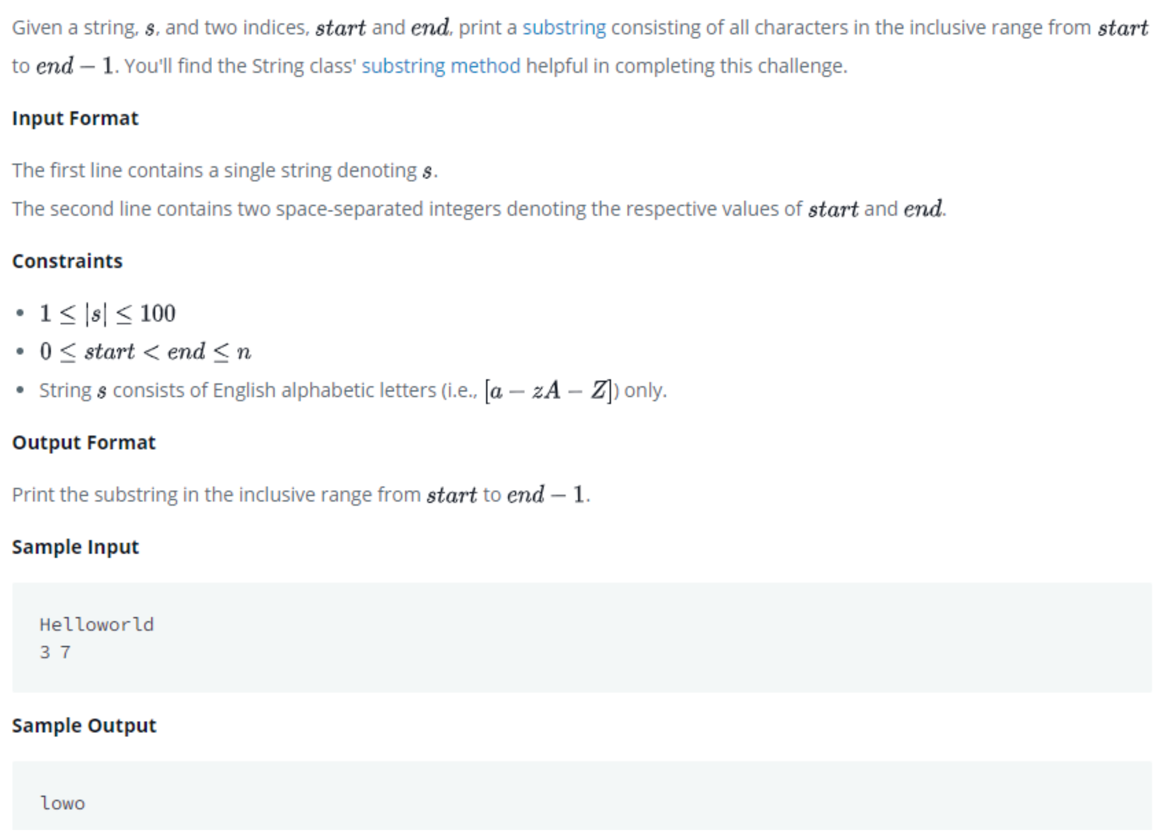
Sample Output

```
Yes
```

Answer

```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        String A= sc.next();
         System.out.println(A.equalsIgnoreCase(new StringBuilder(A).rev
erse().toString())
          ? "Yes" : "No");
    }
}
```

7. Question “Java Substring”



Given a string, s, and two indices, $start$ and end, print a substring consisting of all characters in the inclusive range from $start$ to $end - 1$. You'll find the String class' substring method helpful in completing this challenge.

Input Format

The first line contains a single string denoting s.
The second line contains two space-separated integers denoting the respective values of $start$ and end.

Constraints

- $1 \leq |s| \leq 100$
- $0 \leq start < end \leq n$
- String s consists of English alphabetic letters (i.e., $[a - zA - Z]$) only.

Output Format

Print the substring in the inclusive range from $start$ to $end - 1$.

Sample Input

```
Helloworld
3 7
```

Sample Output

```
lowo
```

Answer

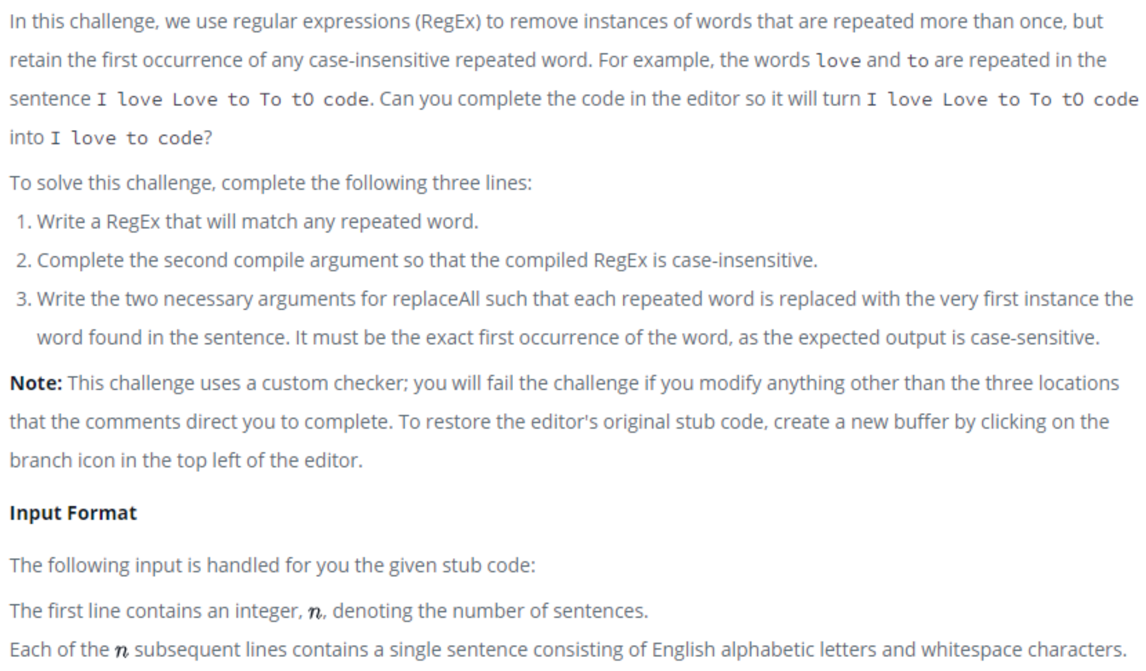
```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String S = in.next();
        int start = in.nextInt();
        int end = in.nextInt();

        System.out.println(S.substring(start, end));
    }
}
```

8.

NAMA : JAMALUDIN

TYPE : MEDIUM

1.Question “Java Regex 2”



In this challenge, we use regular expressions (RegEx) to remove instances of words that are repeated more than once, but retain the first occurrence of any case-insensitive repeated word. For example, the words `love` and `to` are repeated in the sentence `I love Love to To tO code`. Can you complete the code in the editor so it will turn `I love Love to To tO code` into `I love to code`?

To solve this challenge, complete the following three lines:

1. Write a RegEx that will match any repeated word.
2. Complete the second compile argument so that the compiled RegEx is case-insensitive.
3. Write the two necessary arguments for replaceAll such that each repeated word is replaced with the very first instance the word found in the sentence. It must be the exact first occurrence of the word, as the expected output is case-sensitive.

Note: This challenge uses a custom checker; you will fail the challenge if you modify anything other than the three locations that the comments direct you to complete. To restore the editor's original stub code, create a new buffer by clicking on the branch icon in the top left of the editor.

Input Format

The following input is handled for you the given stub code:

The first line contains an integer, n, denoting the number of sentences.
Each of the n subsequent lines contains a single sentence consisting of English alphabetic letters and whitespace characters.

Answer

```java
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class DuplicateWords {

    public static void main(String[] args) {

        String regex = "\\b(\\w+)(?:\\W+\\1\\b)+";
        Pattern p = Pattern.compile(regex,  Pattern.CASE_INSENSITIVE);

        Scanner in = new Scanner(System.in);
        int numSentences = Integer.parseInt(in.nextLine());
        while (numSentences-- > 0) {
            String input = in.nextLine();

            Matcher m = p.matcher(input);

            // Check for subsequences of input that match the compiled
pattern
            while (m.find()) {
                input = input.replaceAll(m.group(), m.group(1) );
            }

            // Prints the modified sentence.
            System.out.println(input);
        }

        in.close();
    }
}
```