

## 1. Roman Numerals

## Problem Statement:

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

## Symbol Value

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

- I can be placed before V (5) and X (10) to make 4 and 9.
- X can be placed before L (50) and C (100) to make 40 and 90.
- C can be placed before D (500) and M (1000) to make 400 and 900.

## Constraints:

- $1 \leq s.length \leq 15$
- s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
- It is guaranteed that s is a valid roman numeral in the range \*1, 3999+.

## Input Format:

Read the string

## Output Format:

Print the numeral equivalent to roman.

## SAMPLE INPUT

Input: s = "III"

Input: s = "LVIII"

Input: s = "MCMXCIV"

SAMPLE OUTPUT

Output: 3 // Explanation: III = 3.

Output: 58 // Explanation: L = 50, V= 5, III = 3

Output: 1994 //Explanation: M = 1000, CM = 900, XC = 90 and IV =

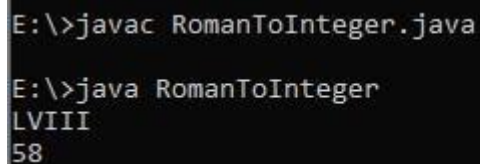
4. import java.util.Scanner;

```
public class RomanToInteger { public
static void main(String[] args) {
Scanner sc = new Scanner(System.in); String
s = sc.next(); int sum = 0;
```

```
for (int i = 0; i < s.length(); i++) { int val1 =
getValue(s.charAt(i)); int val2 = (i + 1 < s.length()) ?
getValue(s.charAt(i + 1)) : 0; if
(val1 < val2) {
sum -= val1;
} else { sum
+= val1;
}
}
```

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

```
public static int getValue(char r) {
if (r == 'I') return 1; if (r == 'V')
return 5; if (r == 'X') return 10; if (r
== 'L') return 50; if (r == 'C') return
100; if (r == 'D') return 500; if (r
== 'M') return 1000; return 0;
}
}
```



```
E:\>javac RomanToInteger.java
E:\>java RomanToInteger
LVIII
58
```

2. Palindrome

### Problem Statement:

A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string *s*, return true if it is a palindrome, or false otherwise.

Input Format: Read  
the sentence

Output Format:  
Print true or false

### SAMPLE INPUT

Input: *s* = "A man, a plan, a canal: Panama"

SAMPLE OUTPUT True

// Explanation: "amanaplanacanalpanama" is a palindrome

Coding: import java.util.Scanner;

```
public class PalindromeCheck { public
static void main(String[] args) {
String s = new Scanner(System.in).nextLine().toLowerCase(); for
(int i = 0, j = s.length() - 1; i < j; i++, j--) { while (i < j &&
!Character.isLetterOrDigit(s.charAt(i))) i++; while (i < j
&& !Character.isLetterOrDigit(s.charAt(j))) j--; if
(s.charAt(i) != s.charAt(j)) {
System.out.println(false); return;
}
}
System.out.println(true);
}
}
```

```
E:\>javac PalindromeCheck.java

E:\>java PalindromeCheck
A man,aplan,a canal:panama
true

E:\>_
```

### 3.Longest common Suffix

#### Problem Statement:

Write a function to find the longest common suffix string amongst an array of strings. If there is no common prefix, return "Not Matching" Constraints:

- $1 \leq \text{strs.length} \leq 200$
- $0 < \text{strs}[i].\text{length} \leq 200$
- $\text{strs}[i]$  consists of only lowercase English letters.

#### Input Format:

Read the array string length Read  
the string elements

#### Output Format:

Print the longest suffix

#### SAMPLE INPUT

```
3
"raining" "singing" " flying"
```

#### SAMPLE OUTPUT

```
"ing"
```

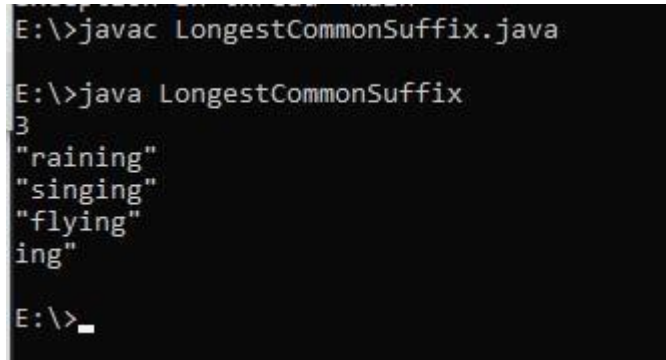
#### Coding:

```
import java.util.Scanner; public class
LongestCommonSuffix { public static
void main(String[] args) { Scanner sc =
new Scanner(System.in); int n =
sc.nextInt(); sc.nextLine();
String[] str = new String[n]; for (int i = 0; i < n;
i++) str[i] = sc.nextLine();
String suffix = str[0]; for (int i = 1; i < n; i++) { int j = suffix.length() - 1,
k = str[i].length() - 1; while (j >= 0 && k >= 0 && suffix.charAt(j)
```

```

== strs[i].charAt(k)) { j--; k--;
}
suffix = suffix.substring(j + 1);
if (suffix.isEmpty()) {
System.out.println("Not Matching"); return;
}
}
System.out.println(suffix);
}
}

```



```

E:\>javac LongestCommonSuffix.java
E:\>java LongestCommonSuffix
3
"raining"
"singing"
"flying"
"ing"
E:\>

```

#### 4. Print even length words Problem

Statement:

Given a string str, write a Java program to print all words with even length in the given string.

Input Format:

First Line : A sentence This is  
an engineering course

Output  
Format: Print the words one  
by one.

is

An course

SAMPLE INPUT

This is a java language

SAMPLE OUTPUT

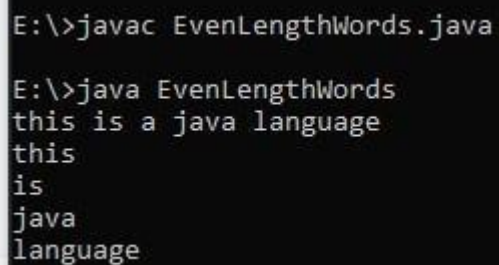
This is

java

language

```
coding: import
java.util.Scanner;
```

```
public class EvenLengthWords {
    public static void main(String[] args) { String str =
        new Scanner(System.in).nextLine(); for
        (String word : str.split(" ")) { if (word.length() %
        2 == 0) {
        System.out.println(word);
        }
        }
    }
}
```



```
E:\>javac EvenLengthWords.java
E:\>java EvenLengthWords
this
is
java
language
```

## 5.StringBuffer

### Problem Statement:

StringBuffer may have characters and substrings inserted in the middle or appended to the end. It will automatically grow to make room for such additions and often has more characters preallocated than are actually needed, to allow room for growth.

Important Constructors of StringBuffer class

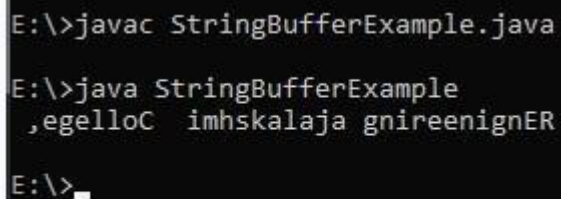
- ☐ **StringBuffer():** creates an empty string buffer with an initial capacity of 16.
- ☐ **StringBuffer(String str):** creates a string buffer with the specified string.
- ☐ **StringBuffer(int capacity):** creates an empty string buffer with the specified capacity as length.

Follow the steps to accomplish the task .

1. Create a string using StringBuffer class. Str = "Rajalakshmi ";
2. Append a string " College, Kanchipuram".
3. Insert a string "Engineering " at index 1.
4. Replace "Kanchipuram" with "Chennai"
5. Delete the string ""Chennai".

7. Print the string in reverse form.

```
public class StringBufferExample { public
static void main(String[] args) {
StringBuffer str = new StringBuffer("Rajalakshmi ");
str.append(" College, Kanchipuram"); str.insert(1, "Engineering
"); int start = str.indexOf("Kanchipuram"); if
(start != -1) { str.replace(start, start + "Kanchipuram".length(), "Chennai");
} start = str.indexOf("Chennai"); if (start != -1) { str.delete(start, start +
"Chennai".length());
}
System.out.println(str.reverse());
}
}
```



```
E:\>javac StringBufferExample.java
E:\>java StringBufferExample
,egelloC imhskalaja gnireenignER
E:\>
```

6.Capitalize the letters Problem

Statement:

You are given a string title consisting of one or more words separated by a single space, where each word consists of English letters. Capitalize the string by changing the capitalization of each word such that:

- If the length of the word is 1 or 2 letters, change all letters to lowercase. • Otherwise, change the first letter to uppercase and the remaining letters to lowercase.

Return the capitalized title.

"Capitalize The Title"// Explanation:

Since all the words have a length of at least 3, the first letter of each word is uppercase and the remaining letters are lowercase.

Input Format: First Line

: A sentence Output

Format:

Print the sentence with first letter capital case

SAMPLE INPUT

capiTalze tHe titLe

SAMPLE OUTPUT

Capitalize The Title

Coding: import java.util.Scanner;

```
public class CapitalizeTitle { public
static void main(String[] args) {
String title = new Scanner(System.in).nextLine();
String[] words = title.split(" ");
StringBuilder capitalizedTitle = new StringBuilder();

for (String word : words) { if (word.length() > 2) {
capitalizedTitle.append(Character.toUpperCase(word.charAt(0))
) .append(word.substring(1).toLowerCase()); } else {
capitalizedTitle.append(word.toLowerCase());
}
capitalizedTitle.append(" ");
}

System.out.println(capitalizedTitle.toString().trim()); }
}
```