

## **LEVEL-1 (EASY)**

### **SET - A**

#### **PROBLEM:**

- **REVERSE A NUMBER WITH CONSTRAINT (NO LEADING ZEROS)**

Write a program that takes a positive integer as input and reverses it, ensuring that there are no leading zeros in the reversed number.

#### ***INPUT:***

*A single positive integer  $n$  ( $1 \leq n \leq 10^9$ ).*

#### ***OUTPUT:***

*The reversed number with no leading zeros.*

#### ***EXAMPLE:***

##### ***INPUT:***

120580120450

##### ***OUTPUT:***

85021

54021

#### **CONSTRAINTS:**

- ❖ The reversed number should not have any leading zeros.
- ❖ For example, if the input is 300, the output should be 3, not 003.

## **LEVEL-1 (EASY)**

### **SET - B**

#### **PROBLEM:**

- **FIND SECOND MAXIMUM AND MINIMUM, AND CHECK EVEN/ODD.**

Write a program that takes an array of positive integers as input and performs the following tasks:

1. Find the second maximum and second minimum numbers in the array.
2. Check whether the second maximum and second minimum numbers are even or odd.

#### **INPUT:**

- *The first line contains an integer  $n$  ( $2 \leq n \leq 100$ ), the number of elements in the array.*
- *The second line contains  $n$  space-separated positive integers, representing the array elements.*

#### **OUTPUT:**

- *The second maximum and second minimum integers separated by a space.*
- *"Even" if the second maximum is even; otherwise, "Odd."*
- *"Even" if the second minimum is even; otherwise, "Odd."*

#### **EXAMPLE:**

##### **INPUT:**

```
6
5 12 3 8 10 66 5 12 3 8 6
```

##### **OUTPUT:**

```
10 5
Even
Odd
```

- In this example, the second maximum is 10 (Even) and the second minimum is 5 (Odd)

## **LEVEL-1(EASY)**

### **SET – C**

#### **PROBLEM:**

- **ALTERNATE LETTER CASE CONVERSION**

Write a program that takes a string as input and converts it into an alternating uppercase and lowercase format, where each character is transformed sequentially.

#### **INPUT:**

*A single string containing alphanumeric characters and symbols. The string length is at most 100 characters.*

#### **OUTPUT:**

*The input string with characters in an alternating uppercase and lowercase format.*

#### **EXAMPLES:**

##### **INPUT:**

World

##### **OUTPUT:**

WoRlD

##### **Input:**

HEllo

##### **Output:**

HeLlO

## **LEVEL-2 (INTERMEDIATE)**

### **SET – A**

#### **PROBLEM:**

- Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of candidates where the chosen numbers sum to target. You may return the combinations in any order.

The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to target is less than 150 combinations for the given input.

#### **EXAMPLE 1:**

##### **INPUT:**

candidates = [2,3,6,7], target = 7

##### **OUTPUT:**

[[2,2,3],[7]]

##### **EXPLANATION:**

2 and 3 are candidates, and  $2 + 2 + 3 = 7$ . Note that 2 can be used multiple times.

7 is a candidate, and  $7 = 7$ .

These are the only two combinations.

## **LEVEL-2 (INTERMEDIATE)**

### **SET – B**

#### **PROBLEM:**

- Replace the characters of a string with the next character of that character in alphabetic order only if the ascii value of that character is prime number.

#### **EXAMPLE 1:**

##### **INPUT:**

Hello, World!

##### **OUTPUT:**

Hello, Xsmle!

##### **EXPLANATION:**

The ASCII values of the characters in "Hello, World!" are as follows:

'H' (72): Not prime, so it remains 'H', 'e' (101): Prime, so it becomes 'f' (102).

'l' (108): Not prime, so it remains 'l', 'l' (108): Not prime, so it remains 'l'.

'o' (111): Not prime, so it remains 'o', ',' (44): Not prime, so it remains ','.

' ' (32): Not prime, so it remains ' '.

'W' (87): Not prime, so it remains 'W'.

'o' (111): Not prime, so it remains 'o'.

'r' (114): Prime, so it becomes 's' (115).

'l' (108): Not prime, so it remains 'l'.

'd' (100): Prime, so it becomes 'e' (101).

'!' (33): Not prime, so it remains '!'.

The modified string is "Hello, Xsmle!" where 'f' and 's' are the next prime ASCII characters for 'e' and 'r' respectively.

## LEVEL-3 (HARD)

### PROBLEM:

- Mr. Vincent works in a door mat manufacturing company. One day, he designed a new door mat with the following specifications:  
Mat size must be  $M \times N$ . ( $N$  is an odd natural number, and  $M$  is 3 times  $N$ .)  
The design should have 'WELCOME' written in the center.  
The design pattern should only use |, . and - characters.

### SAMPLE DESIGN:

Size: 7 x 21

```

-----|.-----
-----|.|.|.|.-----
---|.|.|.|.|.|.|.-----
---WELCOME-----
---|.|.|.|.|.|.|.-----
-----|.|.|.|.-----
-----|.-----

```

Size: 11 x 33

```

-----|.-----
-----|.|.|.|.|.-----
-----|.|.|.|.|.|.|.-----
---|.|.|.|.|.|.|.|.|.|.-----
---WELCOME-----
---|.|.|.|.|.|.|.|.|.|.-----
-----|.|.|.|.|.-----
-----|.-----

```

### ***INPUT FORMAT***

A single line containing the space separated values of `N` and `M`.

### ***CONSTRAINTS***

- $5 < N < 101$
- $15 < M < 303$

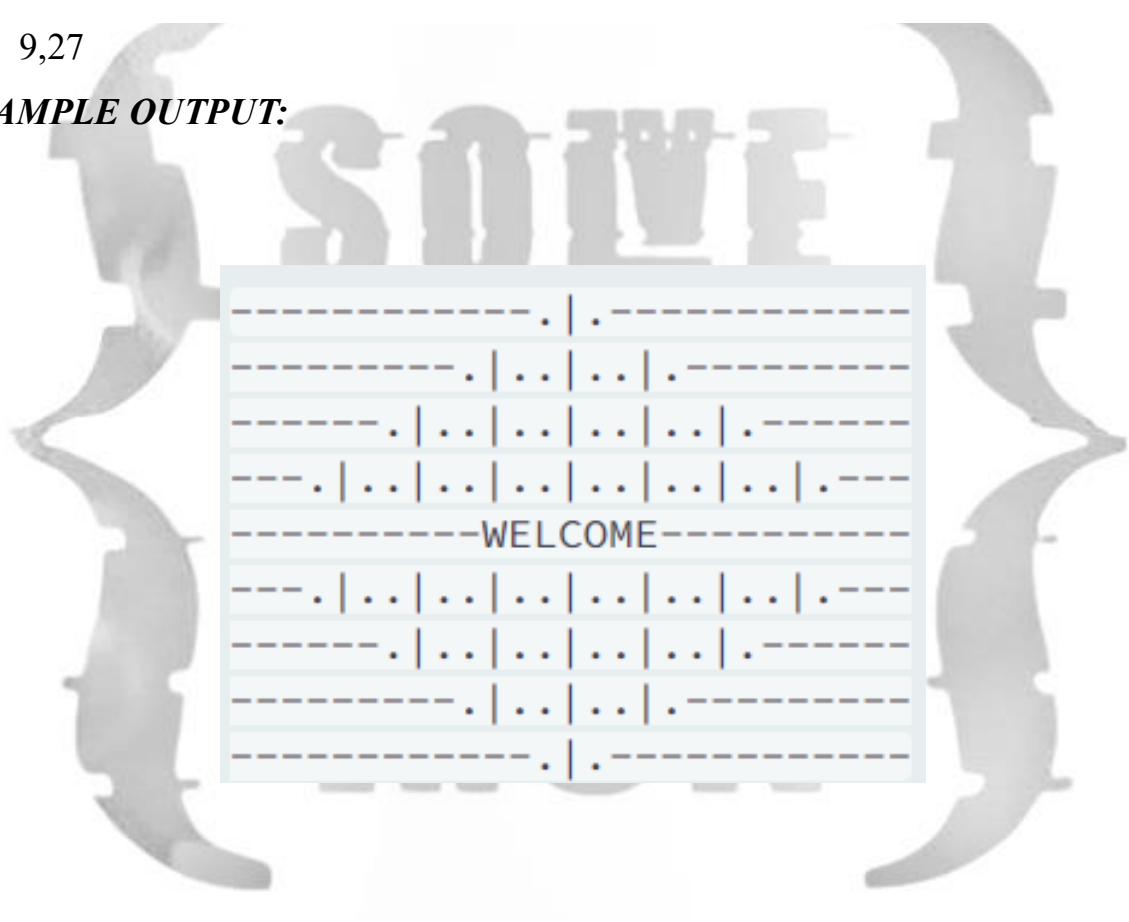
### ***OUTPUT FORMAT***

Output the design pattern.

### ***SAMPLE INPUT***

- 9,27

### ***SAMPLE OUTPUT:***



```

      . | .
-----
    . | . | . | .
-----
  . | . | . | . | . | .
-----
. | . | . | . | . | . | .
-----
      WELCOME
-----
    . | . | . | . | . | . | .
-----
  . | . | . | . | . | .
-----
    . | . | . | .
-----
      . | .
-----

```