Sample

Entry Test

Hanoi, 03/2025

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	07/Apr/2025	Create a newly issue	Create a sample for ET	DieuNT1	

Contents

Entry Test

1. Question 1

Level: Easy-Medium

Title: VNUMBER

Description:

In arithmetic, an abundant number is a number whose sum of divisors (excluding itself) is greater than the number itself. For example, the number 12 has the sum of the divisors (excluding 12) of 1+2+ 3+ 4+6= 16 >12. Hence 12 is an abundant number.

Requirements: Count how many abundant numbers there are in the [L,R] segment.

Data: There is a single line containing two integers L,R (1≤L≤R≤10^5).

Numbers on the same line are separated by at least one space.

Result: A single integer being the number of abundances in the [L,R] segment.

Example:

Input	Output	Explanation
1 50	9	There are 9 abundant number in the [1,50] segment

Function signature (C++):

```
int vnumber(int l, int r) {
    // PUT YOUR CODE HERE
}
```

2. Question 2

Language: English

Difficult: Easy-Medium

Title: Simplify square root

Description: A square root of a positive integer number can be simplified by pulling the square part out.

For example: $\sqrt{12} = 2 * \sqrt{3}$.

Requirements: Given a positive integer n (1 <= n <= 999999). Find the number over the square root sign and the number under the square root sign.

Input: positive integer n (1 <= n <= 999999)

Output: integer array, contains 2 values. The first value is the number over the square root sign, the second value is the number under the square root sign

Note: Any value in output can be 1

Example:

Input	Output	Explanation
12	[2, 3]	12 = 2*2*3 ➔ $\sqrt{12} = 2 * \sqrt{3}$
9	[3, 1]	9 = 3*3*1 ➔ $\sqrt{9} = 3 * \sqrt{1}$
5	[1, 5]	5 = 1*5 ➔ $\sqrt{5} = 1 * \sqrt{5}$
1	[1, 1]	1 = 1*1 ➔ $\sqrt{1} = 1 * \sqrt{1}$

Function signature (C#):

```
int[] SimplifySquareRoot(int n)
{
    //PUT YOUR CODE HERE
}
```

3. Question 3

Difficult: Easy-Medium

Title: Create new account

Description: To create account from full name, we use first name and first letter of middle name and last name (if there). To keep all accounts are unique, we add sequence number at the end of account.

For example: the full name “John Doe” should have account “johnd”. But, if the account “johnd” already exist, the account should be “johnd1”.

Requirements: Given a full name and the array of existing accounts. Your task is to create account for the user.

Input:

string – full name of new user

array – array of string, as array of existing accounts.

Output: string – the new account. All letters are in lower case

Notes: input full name is in format:

- Does NOT contain leading and trailing spaces
- Does NOT contain multiple consecutive spaces
- Does NOT contain special characters
- Capitalize the first letter of each word

Example:

Input	**Output**	**Explanations**
Full name: "John Smith"	johns2	The account "johns" makes from first name and the first letter of last name

Accounts: ["markj", "marryk", "johns", "johns1"]		Use sequence number is 2 because existing "johns" and "johns1"

Function signature (C#):

```
string CreateNewAccount(string fullName, string[] accounts)
{
    //PUT YOUR CODE HERE
}
```

4. Question 4

Difficult: Easy

Title: Odd one out

Requirements: Given list of n positive integers (n >=3) in an array. Most numbers have same sum of digits, except one. Your task is finding the different one.

Input:

array - array of n positive integer numbers. There are at least 3 elements in the array (n>=3) and (n-1) elements have same sum of digits.

Output: integer – number which has different sum of digits to others.

Example:

Input	Output	Explanations
[25, 61, 43, 54, 133]	54	Each 25, 61, 43, 133 has sum of digits is 2 + 5 = 6 + 1 = 4 + 3 = 1 + 3 + 3 = 7 The different one is 54, which has sum of digits is 9

Function signature:

```
int OddOneOut(int[] array)
{
    //PUT YOUR CODE HERE
}
```