

# Rajalakshmi Engineering College

Name: shobbika T  
Email: 240701502@rajalakshmi.edu.in  
Roll no: 240701502  
Phone: 7305423247  
Branch: REC  
Department: I CSE FE  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23221\_Python Programming

### REC\_Python\_Week 6\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

### Section 1 : Coding

#### 1. Problem Statement

Write a program to read the Register Number and Mobile Number of a student. Create user-defined exception and handle the following:

If the Register Number does not contain exactly 9 characters in the specified format(2 numbers followed by 3 characters followed by 4 numbers) or if the Mobile Number does not contain exactly 10 characters, throw an `IllegalArgumentException`. If the Mobile Number contains any character other than a digit, raise a `NumberFormatException`. If the Register Number contains any character other than digits and alphabets, throw a `NoSuchElementException`. If they are valid, print the message 'valid' or else print an Invalid message.

#### ***Input Format***

The first line of the input consists of a string representing the Register number.

The second line of the input consists of a string representing the Mobile number.

### **Output Format**

The output should display any one of the following messages:

If both numbers are valid, print "Valid".

If an exception is raised, print "Invalid with exception message: ", followed by the specific exception message.

Refer to the sample output for the formatting specifications.

### **Sample Test Case**

Input: 19ABC1001  
9949596920

Output: Valid

### **Answer**

```
import re
```

```
class IllegalArgumentException(Exception):  
    pass
```

```
class NumberFormatException(Exception):  
    pass
```

```
class NoSuchElementException(Exception):  
    pass
```

```
try:
```

```
    register_number = input().strip()  
    mobile_number = input().strip()
```

```
    if len(register_number) != 9:  
        raise IllegalArgumentException("Register Number should have exactly 9  
characters.")
```

```
    if not re.match(r"^\d{2}[A-Za-z]{3}\d{4}$", register_number):
```

```

if not re.match(r"^[A-Za-z0-9]+$", register_number):
    raise NoSuchElementException("Register Number should only contain
    alphabets and digits.")
else:
    raise IllegalArgumentException("Register Number should have the format:
    2 numbers, 3 characters, and 4 numbers.")

if len(mobile_number) != 10:
    raise IllegalArgumentException("Mobile Number should have exactly 10
    characters.")

if not mobile_number.isdigit():
    raise NumberFormatException("Mobile Number should only contain digits.")
print("Valid")

except (IllegalArgumentException, NumberFormatException,
NoSuchElementException) as e:
    print(f"Invalid with exception message: {e}")

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

In the enchanted realm of Academia, you, the Academic Alchemist, are bestowed with a magical quill and a parchment to weave the grades of aspiring students into a tapestry of academic brilliance.

The mission is to craft a Python program that empowers faculty members to enter student grades for any two subjects, stores these magical grades in a mystical file, and then, with a wave of your virtual wand, calculates the GPA to unveil the true essence of academic achievement.

### **Input Format**

The input format is a string representing the student's name, any two subjects, and corresponding grades.

After entering grades, they can type 'done' when prompted for the student's name.

### **Output Format**

The output should display the (average of grades) calculated GPA with a precision of two decimal places.

The magical grades will be saved in a mystical file named "magical\_grades.txt".

Refer to the sample output for format specifications.

### **Sample Test Case**

Input: Alice

Math

95

English

88

done

Output: 91.50

### **Answer**

```
with open("magical_grades.txt", "w") as file:
```

```
    while True:
```

```
        name = input().strip()
```

```
        if name.lower() == "done":
```

```
            break
```

```
        subject1 = input().strip()
```

```
        grade1 = int(input().strip())
```

```
        subject2 = input().strip()
```

```
        grade2 = int(input().strip())
```

```
        file.write(f"{name},{subject1},{grade1},{subject2},{grade2}\n")
```

```
    gpa = (grade1 + grade2) / 2
```

```
    print(f"{gpa:.2f}")
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Alice is developing a program called "Name Sorter" that helps users organize and sort names alphabetically.

The program takes names as input from the user, saves them in a file, and then displays the names in sorted order.

File Name: sorted\_names.txt.

#### ***Input Format***

The input consists of multiple lines, each containing a name represented as a string.

To end the input and proceed with sorting, the user can enter 'q'.

#### ***Output Format***

The output displays the names in alphabetical order, each name on a new line.

Refer to the sample output for the formatting specifications.

#### ***Sample Test Case***

Input: Alice Smith

John Doe

Emma Johnson

q

Output: Alice Smith

Emma Johnson

John Doe

#### ***Answer***

```
names = []
```

```
while True:
```

```
    name = input().strip()
```

```
    if name.lower() == 'q':
```

```
        break
```

```
names.append(name)
names.sort()

with open("sorted_names.txt", "w") as file:
    for name in names:
        file.write(name + "\n")
    print(name)
```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

A shopkeeper is recording the daily sales of an item for N days, where the price of the item remains the same for all days. Write a program to calculate the total sales for each day and save them in a file named sales.txt that can store the data for a maximum of 30 days. Then, read the file and display the total earnings for each day.

Note: Total Earnings for each day = Number of Items sold in that day × Price of the item.

##### **Input Format**

The first line of input consists of an integer N, representing the number of days.

The second line of input consists of N space-separated integers representing the number of items sold each day.

The third line of input consists of an integer M, representing the price of the item that is common for all N days.

##### **Output Format**

If the number of days entered exceeds 30 ( $N > 30$ ), the output prints "Exceeding limit!" and terminates.

Otherwise, the code reads the contents of the file and displays the total earnings for each day on separate lines.

Contents of the file: The total earnings for N days, with each day's earnings appearing on a separate line.

Refer to the sample output for the formatting specifications.

**Sample Test Case**

Input: 4  
5 10 5 0  
20  
Output: 100  
200  
100  
0

**Answer**

```
N = int(input().strip())
if N > 30:
    print("Exceeding limit!")
else:
    items_sold = list(map(int, input().strip().split()))
    M = int(input().strip())
    with open("sales.txt", "w") as file:
        for count in items_sold:
            total = count * M
            file.write(str(total) + "\n")
    with open("sales.txt", "r") as file:
        for line in file:
            print(line.strip())
```

**Status :** Correct

**Marks :** 10/10