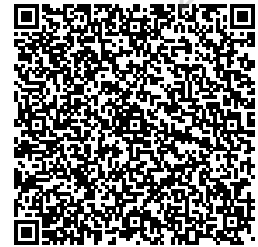


# Rajalakshmi Engineering College

Name: Jovitha J  
Email: 240701219@rajalakshmi.edu.in  
Roll no: 240701219  
Phone: 7825034021  
Branch: REC  
Department: I CSE AH  
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

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**

```
def main():
    names = []

    while True:
        name = input().strip()
        if name.lower() == 'q':
            break
        names.append(name)

    # Sort names alphabetically
    sorted_names = sorted(names)

    # Save to file
    with open("sorted_names.txt", "w") as f:
        for name in sorted_names:
            f.write(name + "\n")

    # Print sorted names (each on a new line)
    for name in sorted_names:
        print(name)

if __name__ == "__main__":
    main()
```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Alex is creating an account and needs to set up a password. The program prompts Alex to enter their name, mobile number, chosen username, and desired password. Password validation criteria include:

Length between 10 and 20 characters. At least one digit. At least one special character from !@#\$%^&\* set. Display "Valid Password" if criteria are met; otherwise, raise an exception with an appropriate error message.

### **Input Format**

The first line of the input consists of the name as a string.

The second line of the input consists of the mobile number as a string.

The third line of the input consists of the username as a string.

The fourth line of the input consists of the password as a string.

### **Output Format**

If the password is valid (meets all the criteria), it will print "Valid Password"

If the password is weak (fails any one or more criteria), it will print an error message accordingly.

Refer to the sample outputs for the formatting specifications.

### **Sample Test Case**

Input: John  
9874563210  
john

john1#nhoj

Output: Valid Password

### **Answer**

# Input

name = input()

mobile = input()

username = input()

password = input()

if not(10 <= len(password) <= 20):

    print("Should be a minimum of 10 characters and a maximum of 20 characters")

elif not any(char.isdigit() for char in password):

    print("Should contain at least one digit")

elif not any(char in "!@#\$%^&\*" for char in password):

    print("It should contain at least one special character")

else:

    print("Valid Password")

**Status : Correct**

**Marks : 10/10**

### **3. 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**

```
# You are using Python
```

```
# Open file to store magical grades  
with open("magical_grades.txt", "w") as file:
```

```
    while True:
```

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

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

```
            break
```

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

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

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

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

```
# Convert grades to integers
```

```
try:
```

```
    grade1 = int(grade1)
```

```
    grade2 = int(grade2)
```

```

except ValueError:
    print("Grades must be numeric.")
    continue

# Validate grade range
if not (0 <= grade1 <= 100 and 0 <= grade2 <= 100):
    print("Grades must be between 0 and 100.")
    continue

# Calculate GPA
gpa = (grade1 + grade2) / 2
print(f"{gpa:.2f}")

# Write to magical file
file.write(f"{name} {subject1} {grade1} {subject2} {grade2}\n")

```

**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())
```

```
if n > 30:
```

```
    print("Exceeding limit!")
```

```
else:
```

```
    items_sold = list(map(int, input().split()))
```

```
    price = int(input())
```

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

```
        for day_sales in items_sold:
```

```
            file.write(str(day_sales * price) + "\n")
```

```
    with open("sales.txt", "r") as file:
```

```
        print(file.read().strip())
```

**Status : Correct**

**Marks : 10/10**