# Rajalakshmi Engineering College

Name: krithika narasimhan
Email: 240701277@rajalakshmi.edu.in
Roll no: 240701277
Phone: 9677451731
Branch: REC
Department: I CSE FC
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 name_sorter():
    # Get names from the user
    names = []
    while True:
        name = input()
        if name.lower() == 'q':
            break
        names.append(name)

    # Sort the names alphabetically
    sorted_names = sorted(names)

    # Save the sorted names to a file
    with open('sorted_names.txt', 'w') as file:
        for name in sorted_names:
            file.write(name + '\n')

    # Display the sorted names
    # print("Sorted names:")
    for name in sorted_names:
        print(name)
```

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

***Status :*** Correct ***Marks : 10/10***

2. 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())
items_input = input().split()
price = int(input())

if N > 30:
    print("Exceeding limit!")
else:
    items = [int(item) for item in items_input]

    with open("sales.txt", "w") as of:
        for item in items:
            of.write(str(item * price) + '\n')

    with open("sales.txt", "r") as inputFile:
        for line in inputFile:
            print(int(line))
```

***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***

```
# Academic Alchemist's Python Spell

magical_grades = {}

while True:
    student_name = input()
    if student_name.lower() == 'done':
        break

    subjects_grades = {}
```

```
    for _ in range(2):
        subject = input()
        if subject.lower() == 'done':
            break

        grade_input = input()
        grade = float(grade_input) if grade_input.replace('.', '', 1).isdigit() else -1

        while not (0 <= grade <= 100):
            grade_input = input()
            grade = float(grade_input) if grade_input.replace('.', '', 1).isdigit() else -1

        subjects_grades[subject] = grade

    magical_grades[student_name] = subjects_grades

# Calculate GPA
total_gpa = sum(sum(grades.values()) for grades in magical_grades.values())
total_subjects = sum(len(grades) for grades in magical_grades.values())
gpa = total_gpa / total_subjects if total_subjects != 0 else 0

# Save magical grades in a mystical file
with open("magical_grades.txt", "w") as file:
    for student, grades in magical_grades.items():
        file.write("{}: {}\n".format(student, grades))
# Display GPA
print("{:.2f}".format(gpa))
```

***Status :*** Correct **Marks : 10/10**

4. Problem Statement

Implement a program that checks whether a set of three input values can form the sides of a valid triangle. The program defines a function is_valid_triangle that takes three side lengths as arguments and raises a ValueError if any side length is not a positive value. It then checks whether the sum of any two sides is greater than the third side to determine the validity of the triangle.

***Input Format***

The first line of input consists of an integer A, representing side1.

The second line of input consists of an integer B, representing side2.

The third line of input consists of an integer C, representing side3.

***Output Format***

The output prints either "It's a valid triangle" if the input side lengths form a valid triangle,

or "It's not a valid triangle" if they do not.

If there is a ValueError, it should print "ValueError: <error_message>".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 3
4
5
Output: It's a valid triangle

***Answer***

```
def is_valid_triangle(side1, side2, side3):
    if side1 <= 0 or side2 <= 0 or side3 <= 0:
        raise ValueError("Side lengths must be positive")

    if side1 + side2 > side3 and side2 + side3 > side1 and side1 + side3 > side2:
        return True
    else:
        return False


def main():
    side1 = int(input())
    side2 = int(input())
    side3 = int(input())
```

```
    try:
        if is_valid_triangle(side1, side2, side3):
            print("It's a valid triangle")
        else:
            print("It's not a valid triangle")
    except ValueError as e:
        print("ValueError: " + str(e))


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

***Status :*** Correct **Marks : 10/10**