# Rajalakshmi Engineering College

Name: Paarthiv suriya  sundaram nagarajan
Email: 240701376@rajalakshmi.edu.in
Roll no: 240701376
Phone: 9445142850
Branch: REC
Department: I CSE FD
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 : 36.5

## Section 1 : Coding

1. 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 !@#$%^&amp;* 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***

```
import re
def validate_password(password):
   if not re.search(r"\d", password):
      raise Exception("Should contain at least one digit")
   if not re.search(r"[!@#$%^&*]", password):
      raise Exception("It should contain at least one special character")
   if not (10 <= len(password) <= 20):
      raise Exception("Should be a minimum of 10 characters and a maximum of
20 characters")
   return "Valid Password"
if __name__ == "__main__":
   name = input()
   mobile_number = input()
   username = input()
   password = input()
   try:
      result = validate_password(password)
      print(result)
   except Exception as e:
    print(e)
```

***Status :*** Partially correct **_Marks : 6.5/10_**

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*

```
# You are using Python
with open("magical_grades.txt", "w") as file:
    while True:
        name = input()
        if name.lower() == "done":
            break
        subject1 = input()
        grade1 = int(input())
        subject2 = input()
        grade2 = int(input())
        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

Write a program to obtain the start time and end time for the stage event show. If the user enters a different format other than specified, an exception occurs and the program is interrupted. To avoid that, handle the exception and prompt the user to enter the right format as specified.

Start time and end time should be in the format 'YYYY-MM-DD HH:MM:SS'If the input is in the above format, print the start time and end time.If the input does not follow the above format, print "Event time is not in the format "

*Input Format*

The first line of input consists of the start time of the event.

The second line of the input consists of the end time of the event.

*Output Format*

If the input is in the given format, print the start time and end time.

If the input does not follow the given format, print "Event time is not in the format".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2022-01-12 06:10:00
2022-02-12 10:10:12

Output: 2022-01-12 06:10:00
2022-02-12 10:10:12

***Answer***

```
# You are using Python
from datetime import datetime
start_time = input()
end_time = input()
try:
    start_dt = datetime.strptime(start_time, "%Y-%m-%d %H:%M:%S")
    end_dt = datetime.strptime(end_time, "%Y-%m-%d %H:%M:%S")
    print(start_time)
    print(end_time)
except ValueError:
    print("Event time is not in the format")
```

***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*

```
# You are using Python
N = int(input())
if N > 30:
   print("Exceeding limit!")
```

```
    exit()
items_sold = list(map(int, input().split()))
M = int(input())
earnings = [items * M for items in items_sold]
with open("sales.txt", "w") as f:
    for e in earnings:
        f.write(str(e) + "\n")
with open("sales.txt", "r") as f:
    for line in f:
        print(line.strip())
```

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