Rajalakshmi Engineering College

Name: Arjun S
Email: 240701048@rajalakshmi.edu.in
Roll no: 240701048
Phone: 9840235318
Branch: REC
Department: I CSE FA
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 : 26.5

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 !@#$%^&* 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

```
n=input()
no = int(input())
u=input()
p=input()
try:
    for i in p:
        if i.isdigit():
            break
    else:
        raise Exception("Should contain at least one digit")

    for i in p:
        if i in "!@#$%^&*":
            break
    else:
        raise Exception("It should contain at least one speical character")

    if len(p)<10 or len(p)>20:
        raise Exception("Should be a minimum of 10 characters and a maximum of
```

```
20 characters")
except Exception as e:
    print(e)
else:
    print("Valid Password")
```

Status : Partially correct **Marks : 6.5/10**

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

```
def record_sales():
    N=int(input())

    if N<15 or N>30:
        print("Exceeding limit!")

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

    while len(items_sold)<N:
        items_sold.append(0)

    M=int(input())

    if M<15 or M>200:
        print("Invalid item price!")
        return

    for i in items_sold:
        if i<0 or i>50:
            print("Invalid number of items sold")
            return

    total_earnings=[items*M for items in items_sold]

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

```
        for earnings in total_earnings:
            file.write(f"{e}\n")

    with open("sales.txt","r") as file:
        for line in file:
            print(line.strip())

record_sales()
```

Status : Wrong ***Marks : 0/10***

3. 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(a,b,c):
    if a<=0 or b<=0 or c<=0:
        raise ValueError("Side lengths must be positive")
    if a+b>c and a+c>b and b+c>a:
        return True
    else:
        return False

try:
    A=int(input())
    B=int(input())
    C=int(input())

    if is_valid_triangle(A,B,C):
        print("It's a valid triangle")
    else:
        print("It's not a valid triangle")

except ValueError as e:
    print(f"ValueError: {e}")
```

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

4. 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

```
from datetime import datetime

try:
    st=input()
    et=input()

    datetime.strptime(st,"%Y-%m-%d %H:%M:%S")
    datetime.strptime(et,"%Y-%m-%d %H:%M:%S")

    print(st)
    print(et)

except ValueError:
    print("Event time is not in the format")
```

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