Rajalakshmi Engineering College

Name: Jenell S G
Email: 240701212@rajalakshmi.edu.in
Roll no: 2116240701212
Phone: 7418493255
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

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

```
# You are using Python
def is_valid_triangle(a, b, c):
    # Check for positive side lengths
    if a <= 0 or b <= 0 or c <= 0:
        raise ValueError("Side lengths must be positive")

    # Check triangle inequality
    if a + b > c and a + c > b and b + c > a:
        return True
    else:
        return False

# Read input values
try:
    side1 = int(input())
    side2 = int(input())
    side3 = int(input())

    # Check if the triangle is valid
    if is_valid_triangle(side1, side2, side3):
        print("It's a valid triangle")
```

```
    else:
        print("It's not a valid triangle")

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

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

2. 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

```
# You are using Python
def name_sorter():
    filename = "sorted_names.txt"
    names = []


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

    # Sort names alphabetically
    names.sort()

    # Write sorted names to the file
    with open(filename, 'w') as file:
        for name in names:
            file.write(name + '\n')

    # Read and print sorted names
    with open(filename, 'r') as file:
        sorted_names = file.readlines()
        for name in sorted_names:
            print(name.strip())

# Run the program
name_sorter()
```

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

3. 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
def record_sales():
    # Read number of days
    N = int(input())

    # Check the limit
    if N > 30:
        print("Exceeding limit!")
        return

    # Read number of items sold each day
    items_sold = list(map(int, input().split()))

    # Read item price
    M = int(input())

    # Calculate total earnings for each day
    earnings = [sold * M for sold in items_sold]

    # Write to sales.txt
    with open("sales.txt", "w") as file:
        for amount in earnings:
            file.write(str(amount) + '\n')

    # Read and print earnings from the file
    with open("sales.txt", "r") as file:
        for line in file:
            print(line.strip())

# Run the program
record_sales()
```

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

```
# You are using Python
from datetime import datetime

def get_event_times():
    try:
        # Read input
        start_time_input = input()
        end_time_input = input()

        # Validate format and correctness
        datetime.strptime(start_time_input, '%Y-%m-%d %H:%M:%S')
```

```
        datetime.strptime(end_time_input, '%Y-%m-%d %H:%M:%S')

        # If no exception, print times
        print(start_time_input, end_time_input)

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

# Run the function
get_event_times()
```

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