**Symbiosis Institute of Technology, Nagpur**

**CA-II GenAI**

**Sub: GenAI Sem: VII**

**Name: Aman Agrawal PRN: 21070521101**

**Q5 Generate a model for Covid 19 with symptoms of parameters like fever, cold, shivering, weight loss, generate 100 model data with random values for each parameter and order by parameter lowest to highest in display based on the input parameter.**

**Ans :** the task is to simulate a Covid-19 symptom tracking system by generating random data for 100 cases. Here's the step-by-step breakdown:

1. A Covid19Case class is defined to represent each case, with attributes for symptoms such as fever, cold, shivering, and weight loss.
2. We generate 100 cases using Python's random module, which assigns random values for each symptom. For instance, fever is a random float between 97.0 and 105.0°F, while cold and shivering are random integers on a scale from 0 to 10. Weight loss is a random float between 0 and 10 kg.
3. The user can select any symptom to sort the cases by (e.g., sort by fever, cold severity, etc.).
4. Finally, the code displays the sorted cases in ascending order based on the chosen symptom.

import random

class Covid19Case:

    def \_\_init\_\_(self, case\_id, fever, cold, shivering, weight\_loss):

        self.case\_id = case\_id

        self.fever = fever

        self.cold = cold

        self.shivering = shivering

        self.weight\_loss = weight\_loss

    def \_\_repr\_\_(self):

        return (f"Case {self.case\_id}: Fever = {self.fever}, Cold = {self.cold}, "

                f"Shivering = {self.shivering}, Weight Loss = {self.weight\_loss}")

def generate\_random\_cases(num\_cases=100, seed\_value=1234):

    cases = []

    random.seed(seed\_value)

    for i in range(1, num\_cases + 1):

        fever = random.uniform(97.0, 105.0)

        cold = random.randint(0, 10)

        shivering = random.randint(0, 10)

        weight\_loss = random.uniform(0.0, 10.0)

        case = Covid19Case(case\_id=i, fever=fever, cold=cold, shivering=shivering, weight\_loss=weight\_loss)

        cases.append(case)

    return cases

def sort\_cases\_by\_symptom(cases, symptom):

    if symptom == "fever":

        return sorted(cases, key=lambda case: case.fever)

    elif symptom == "cold":

        return sorted(cases, key=lambda case: case.cold)

    elif symptom == "shivering":

        return sorted(cases, key=lambda case: case.shivering)

    elif symptom == "weight\_loss":

        return sorted(cases, key=lambda case: case.weight\_loss)

    else:

        raise ValueError("Invalid symptom. Choose from: 'fever', 'cold', 'shivering', 'weight\_loss'.")

if \_\_name\_\_ == "\_\_main\_\_":

    num\_cases = 100

    cases = generate\_random\_cases(num\_cases)

    symptom = input("Enter symptom to sort by (fever, cold, shivering, weight\_loss): ").strip().lower()

    try:

        sorted\_cases = sort\_cases\_by\_symptom(cases, symptom)

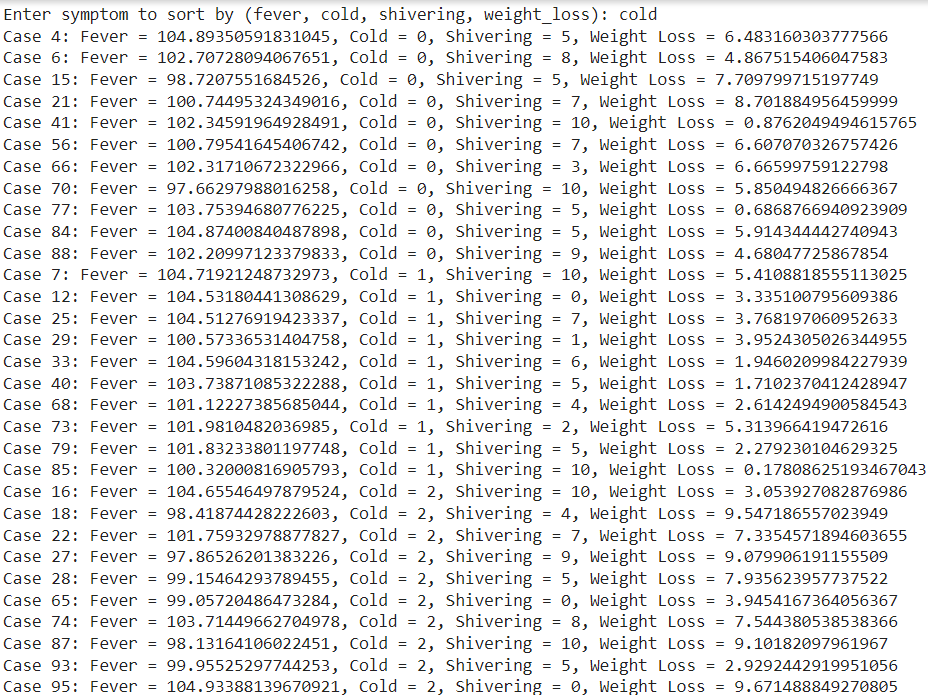
        for case in sorted\_cases:

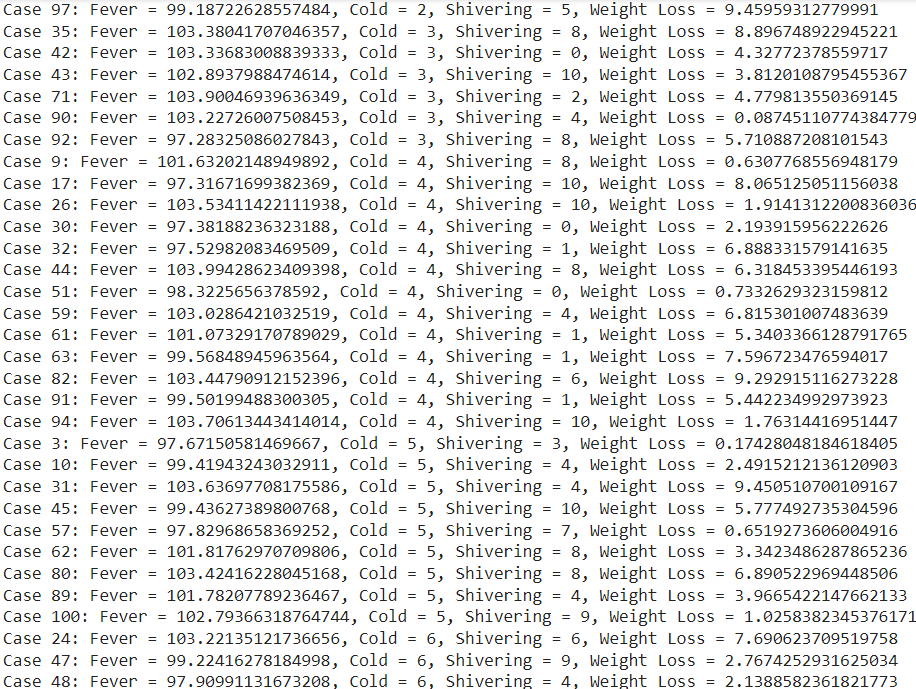
            print(case)

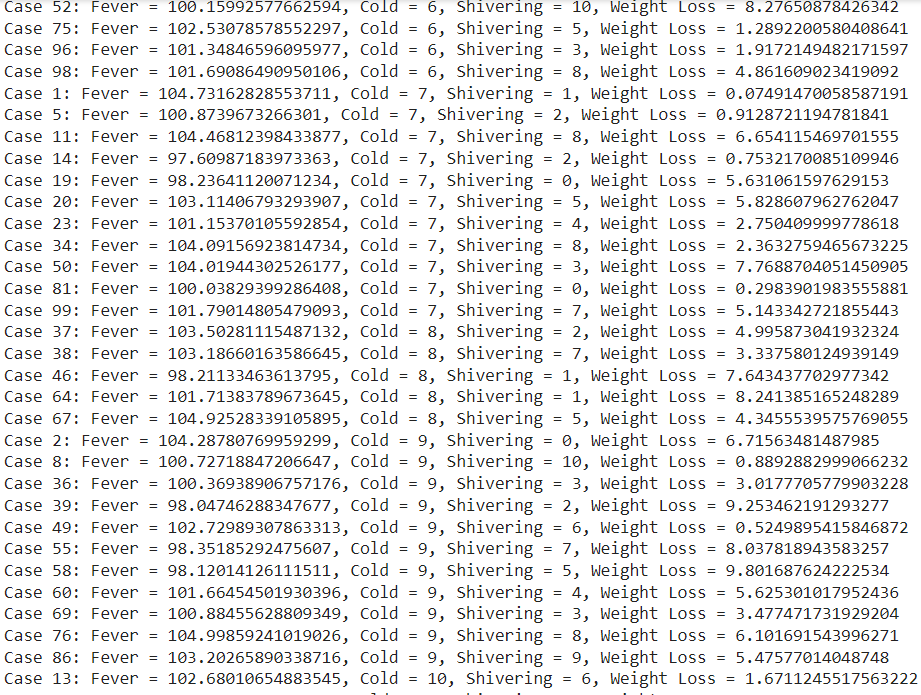
    except ValueError as e:

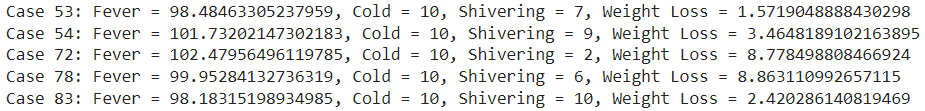
        print(e)

**Output:**

****

****

****

****

**Q6 Generate a model to represent a mathematical equation, write a program to parse the equation, and ask for input for each parameter.**

**Ans**: the task is to parse a mathematical equation and compute the result based on user input for each variable. Here's the breakdown:

1. The program accepts a mathematical equation as a string (e.g., a + b \* c).
2. It identifies the variables in the equation using regular expressions.
3. The user is prompted to input values for each identified variable.
4. The program substitutes the values into the equation, replacing the variables with their respective values.
5. The final equation is evaluated using Python's eval() function to compute the result.
6. The result is then displayed to the user.

import re

def parse\_equation(equation):

    variables = re.findall(r'[a-zA-Z]+', equation)

    return list(set(variables))

def get\_variable\_values(variables):

    values = {}

    for var in variables:

        values[var] = float(input(f"Enter the value for {var}: "))

    return values

def evaluate\_equation(equation, values):

    for var, val in values.items():

        equation = equation.replace(var, str(val))

    return eval(equation)

def main():

    equation = input("Enter the mathematical equation (e.g., a + b \* c): ")

    variables = parse\_equation(equation)

    values = get\_variable\_values(variables)

    result = evaluate\_equation(equation, values)

    print(f"The result of the equation '{equation}' is: {result}")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Output:

