**Question-1)Problem Statement**

You are required to create a program that can perform basic arithmetic operations on complex numbers. A complex number consists of a real part and an imaginary part. The program should provide the following functionality:

1. Create two instances of the **Complex** class using the **default constructor**, Complex c1 and Complex c2.
2. Prompt the user to enter the real and imaginary parts for both c1 and c2 using the **setvalue()** method.
3. Calculate the sum of c1 and c2 using the **sum()** method and store the result in a third instance of the Complex class, c3.
4. Display the result of the addition using the **display()** method.
5. Ensure that a **destructor** is called for each Complex object when it is destroyed.

**Company Tags:** Wipro

**Input format :**

The input consists of two lines, consisting of the real and imaginary real parts of two complex numbers as float values, separated by space.

**Output format :**

The output displays "Sum of two complex numbers is X +Yi" where X is the real part and Y is the imaginary part of the result value as float values rounded to one decimal value.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

In this scenario, the test cases fall under the following constraints:

0.0 ≤ real, imaginary parts ≤ 1000.0

**Sample test cases :**

**Input 1 :**

2.7 3.5

3.2 1.6

**Output 1 :**

Sum of two complex numbers is 5.9 + 5.1i

**Input 2 :**

1.0 2.0

3.0 2.0

**Output 2 :**

Sum of two complex numbers is

**Question-2)Problem Statement**

Create a **Vehicle** class that uses a parameterized constructor to initialize the registration number, make, model, and year of manufacture. Implement a method called displayDetails to print these attributes. Ensure that the class can handle string inputs for the vehicle details.

**Input format :**

The input consists of four lines, each containing:

* A string representing the registration number.
* A string representing the make of the vehicle.
* A string representing the model of the vehicle.
* An integer representing the year of manufacture.

**Output format :**

The output prints the following, each containing the details of a vehicle in the following format:

Registration Number: [registration number]

Make: [make]

Model: [model]

Year of Manufacture: [year of manufacture]

**Refer to the sample output for the exact format.**

**Code constraints :**

In this scenario, the test cases fall under the following constraints:

1900 ≤ year of manufacture ≤ 2100

**Sample test cases :**

**Input 1 :**

KA-01-HH-1234

Toyota

Corolla

2022

**Output 1 :**

Registration Number: KA-01-HH-1234

Make: Toyota

Model: Corolla

Year of Manufacture: 2022

**Question-3)Problem Statement**

Create a program that manages distances using the **Distance** class. The program should allow you to input two distances in feet and inches, and it will calculate their sum. The Distance class contains methods for setting distances, adding distances, and displaying distances. The program will create two Distance objects: d1 and d2, where d2 is a copy of d1 using a copy constructor. It then calculates the sum of d1 and d2 and displays the result.

**Note:** 1 foot = 12 inches

**Input format :**

The first line should contain two integers representing the feet and inches of the first distance, separated by a space.

The second line should contain two integers representing the feet and inches of the second distance, also separated by a space.

**Output format :**

The output will display the sum of the two distances in the format X'Y", where X is the total feet and Y is the total inches.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

In this scenario, the test cases fall under the following constraints:

1 ≤ feet ≤ 100

1 ≤ inches ≤ 12

**Sample test cases :**

**Input 1 :**

10 1

20 1

**Output 1 :**

30'2"

**Input 2 :**

20 12

20 1

**Output 2 :**

41'1"

**Question-4)Problem Statement**

Dhivakar is working on a program that involves processing numerical data. He needs to write a program that allows the user to input a set of numbers, stores them in an array, and then write the numbers to a file named **"sample.dat".** Additionally, the program should find the maximum and minimum values from the input numbers and display them on the console.

**Input format :**

The first line of the input consists of an integer N, representing the number of values to be entered.

The second line of input consists of N space-separated values.

**Output format :**

The output prints two space-separated integers representing the maximum and minimum values, respectively.

**Refer to the sample output for the formatting specifications.**

**Code constraints :**

2 ≤ N ≤ 100

**Sample test cases :**

**Input 1 :**

5

3 5 2 4 1

**Output 1 :**

5 1

**Input 2 :**

3

67 54 3

**Output 2 :**

67 3

**Question-5)Problem Statement**

Krish is working on a C++ program to manage an array of integers. His approach is as follows: Prompt the user to input an integer n, followed by n integer values. Store these values in a binary file named "**numbers.dat**" separated by spaces. Then, read the contents of "numbers.dat" and print them to the console.

**Input format :**

The first line consists of an integer n.

The second line consists of n space-separated integers.

**Output format :**

The program displays the integers stored in the binary file, each separated by a space.

If n < 0 or n > 100, the output prints "Invalid entry".

**Code constraints :**

1 <= n <= 100

**Sample test cases :**

**Input 1 :**

-1

**Output 1 :**

Invalid entry

**Input 2 :**

200

893 267 142 5 686 743 517 506 21 493 671 22 304 540 788 116 484 505 810 318 340 755 599 66 775 793 884 306 140 692 435 680 273 136 349 229 964 32 960 300 726 277 958 500 525 565 363 918 628 667 678 453 393 129 569 627 888 252 467 332 683 54 436 934 339 494 224 729 190 222 837 598 180 417 396 554 429 376 414 197 985 477 877 489 102 71 236 836 65 740 718 582 817 365 778 112 7 178 564 699 590 584 991 44 186 465 439 244 430 925 103 682 219 130 732 470 191 479 241 503 736 629 246 911 796 425 571 324 557 196 386 316 488 617 491 373 381 73 819 445 711 892 643 251 968 509 128 860 745 679 80 928 606 504 501 560 413 938 710 468 204 294 717 87 357 423 358 105 371 933 398 833 639 859 645 207 721 61 741 872 424 157 961 416 869 940 122 48 158 549 223 462 422 551 994 151 317 360 174 183

**Output 2 :**

Invalid entry

**Input 3 :**

5

-71 278 3 425 50

**Output 3 :**

-71 278 3 425 50