OOAIA Lab 6

*submit the design document by 4:50 pm today.

Jack Reacher, an ex-major in the U.S. military police, gets off a bus in Margrave. As trouble always seems to find him, he is soon arrested by the Margrave Police Department for a crime he didn't commit. Later, he is released after proving his innocence. He discovers that an illegal U.S. dollar counterfeiting operation is behind the crime. Reacher has one principle: *do the right thing* so he decides to fight against the counterfeit operation. Since each dollar bill has a serial number, Reacher needs a device that can perform specific numerical calculations. The specifications of the device are given below.

Device Specifications:

The functionality of the required device can be implemented mainly using four classes, details of which are mentioned below. You may create additional classes, overload existing class methods, and add extra data members in a class as needed.

1) **CurrencySorter**:
    - Define a functor that takes an array of integers, where each integer represents a serial number and sorts that array in-place using quick sort.Do not use any built-in sort function,use your own implementation of quick sort.

2) **FibonacciGenerator**:
    - Define a functor that takes a number i and returns F_i where F_i is the i^{th} number in fibonacci sequence.

    A fibonacci sequence is defined by the recurrence relation $F_i=F_{i-1}+F_{i-2}$. {0,1,1,2,.....} is a fibonacci sequence, here $F_0=0$, $F_1=1$, $F_2=1$ and so on.

3) **PrimeCalculator**:
    - Implement a function **printPrimes** that takes 2 numbers L, R as parameters and prints all the prime numbers that are present in the range [L,R].

- Implement a function **printPrimeSum** that takes 2 numbers L , R as parameters and returns the sum of all prime numbers that are present in the range [L,R].

An Integer X > 1 is said to be prime if it has exactly 2 distinct divisors, 1 and X.

4) **NumberAnalyzer**:
   - Implement a function **isSquareFree** that takes a number X as a parameter and returns true if it is square free, else returns false.

     A number X is said to be square-free if X is not divisible by any perfect square other than 1. For example, 15 is square-free, but 24 is not a square-free number because it is divisible by 4 (a perfect square).

   - Implement a function **countDivisors** that takes a number X as a parameter and returns the total number of divisors of X.

     For example, consider X=12, the divisors of 12 are 1,2,3,4,6,12 so countDivisors(12) will return 6.

   - Implement a function **sumOfDivisors** that takes a number X as a parameter and returns the sum of all divisors of X.

     For example, consider X=12, the divisors of 12 are 1,2,3,4,6,12 so sumOfDivisors(12) will return 28.

Test Cases Format :

In this challenge, there are four types of test cases. The first line of each test case indicates the type of test case. Each type of test case checks the correctness of the functions present in a single class. The format of the different types of test cases is given below.

Type 1 :
This type of test case will check the functionality of the **CurrencySorter** class.

Input Format :
- First line contains the number **1**

- Next line contains an integer **t** ($1 <= t <= 10^3$) - number of operations to be performed.
- Each operation contains 2 lines
    - First line contains an integer **n** ($1 <= n <= 10^6$) - size of the array
    - Next line contains n distinct integers **a_1 a_2 .. a_i ... a_n** ($0 <= a_i <= 10^9$) - a_i represents the serial number of i^{th} currency note.

  For each test case of this type, it is guaranteed that the sum of **n** over all operations will not exceed 10^6.

Output Format :
For every operation print the sorted array.

Type 2 :
This type of test case will check the functionality of the **FibonacciGenerator** class.

Input Format :
- First line contains the number **2**
- Next line contains an integer **t** ($1 <= t <= 2*10^5$) - number of operations to be performed.
- First line of each operation contains an integer **i** ($1 <= i <= 10^{18}$)

Output Format :
- For each operation print F_i modulo 10^9+7. Where F_i is the i^{th} number in the fibonacci sequence.

Type 3 :
This type of test case will check the functionality of the **PrimeCalculator** class.

Input Format :
- First line contains the number **3**.
- Next line contains an integer **t** ($1 <= t <= 10^3$) - number of operations to be performed.
- The next t lines can be any one of the following
    - **printPrimes L R** ($L <= R$, $0 <= L,R <= 10^{12}$, $R-L+1 <= 10^6$) - prints all prime numbers in the range [L,R] separated by a single blank space.If there are no prime numbers in that range, print an empty line.
    - **printPrimeSum L R** ($L <= R$, $0 <= L,R <= 10^{12}$, $R-L+1 <= 10^6$) - prints the sum of all prime numbers in the range [L,R].

For each test case it is guaranteed that the sum of (R-L+1) over all operations will not exceed 10^6.

Output Format :

Print the output of each operation in a new line.

Type 4 :

This type of test case will check the functionality of the **NumberAnalyzer** class.

Input Format :

- First line contains the number **4**.
- Next line contains an integer **t** ($1 <= t <= 10^5$) - number of operations to be performed.
- The next t lines can be any one of the following
    - **isSquareFree X** ($2 <= X <= 10^8$) - prints **yes** if X is square-free else prints **no.**
    - **countDivisors X** ($2 <= X <= 10^8$) - prints the number of divisors of X.
    - **sumOfDivisors X** ($2 <= X <= 10^8$) - prints the sum of divisors of X.

Output Format :

Print the output of each operation in a new line.

Sample Test cases

Test case 1:

Input

1
1
6
55 34 36 46 100 72

Output

34 36 46 55 72 100

Test case 2:

Input

2
3
2
4
6

Output
1
3
8

Test case 3
Input
3
2
printPrimes 1 10
printPrimeSum 1 15
Output
2 3 5 7
41

Test case 4
Input
4
3
isSquareFree 6
countDivisors 24
sumOfDivisors 12
Output
yes
8
28