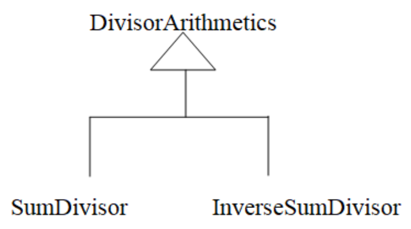
<u>Major Evaluation 2</u>

1. Define an Interface **DivisorArithmetics** with a method ***divisorSum()***. Implement the hierarchy as shown in the following diagram:



- The class **SumDivisor** *implements* the class **DivisorArithmetics.** The method ***divisorSum()*** in **SumDivisor** takes an integer **N** as input and returns the sum of all its divisors.
- The class **InverseSumDivisor** *implements* the class **DivisorArithmetics.** The method ***divisorSum()*** in **InverseSumDivisor** takes an integer **N** as input and returns the sum of the inverse of all the divisors of **N**. (Note: you have to compute sum of divisors first and then divide it by N)
- The class **TestDivisorArithmetics** uses the classes **SumDivisor** and **InverseSumDivisor** to test the functionality of the two classes.

<u>Input Format</u>

- A positive Integer **N.** If N is zero, then print "ZEROINVALID"

Output <u>Format</u>

- First line prints the sum of all the divisors of N.
- Second line prints the sum of inverse of all the divisor of N (rounded Upto Two Decimal places)

<u>Constraint</u>

N is a positive integer

<u>Sample Output/Test Case</u>

Input

6

Output

12

2

Explanation

N= 6

Divisors = {1, 2, 3, 6}

Sum of Divisor = 1+2+3+6 =12

Inverse sum of Divisor = sum of divisor / N =(1 + 2 + 3 + 6)/6 =12/6 =2

Q2. Create a class **Student** with the following members:

- ***rollno*** (String)
- ***sgpa*** (List of four integer elements, represent sgpa for four semesters)
- ***computeCgpa*( )** :- returns the cgpa (Formulae : cgpa = Sum of sgpa/4). If the calculated cgpa is a floating point value, then return the floor value.

Create a class **Teacher** with the following members:

- ***qualification*** (String)
- ***designation*** (String)
- ***computeExperience( )*** :- returns the experience in number of days. Experience calculated from the date of join to 27-09-2021 (including both start and end date).

Class **Teacher** inherits from class **Employee** which has the attributes ***empId*** (Integer) and ***dateOfJoin*** (String in dd-mm-yyyy format). There is an abstract class **Person** with an abstract method ***bioDataView()*** and an attribute **name**(String)**. Student** and **Employee** inherits from **Person** class**.** Use the method ***bioDataView()*** to display the biodata of the **Teacher** and **Student** accordingly (mentioned in output format).

Input Format

The input line is given as a space separated list in the format:

<Character> <Parameters>

<Character> can be (S or T or q)

- ***S*** *: If the <Character> is* ***S*** *then:* <Parameters> is a space separated list where the first parameter is the ***name*** *of the student (only the first name),* the second parameter is the ***rollno*** *of the student* and the third parameter is *the* ***sgpa*** *for four semesters* (space separated).

- ***T*** : *If the <Character> is* ***T*** *then* <Parameters> is a space separated list where the first parameter is the ***name*** *of the employee (only first name)* , the second parameter is the ***empId*** and the third parameter is the ***dateOfJoin*** in dd-mm-yyyy format, fourth parameter is the ***qualification*** *of the teacher(single word)* and the fifth parameter is the ***designation*** *of the teacher*.

- q : Exit the program

Output Format

- If the <Character> in input is **S,** display the biodata of the student in the following format:-

  Name:<name of the student>

  RollNumber:<roll number of the student>

  CGPA:<cgpa of the student> (calculated using **computeCgpa( )**).

- If the <Character> in input is **T** then, display the biodata of the teacher which has following format
  Name:<name of the teacher>
  EmpID:<emp_Id of the teacher>
  Qualification:<Qualification of the teacher>
  Designation:<Designation of the teacher>
  Experience:<Experience of the teacher in ***number of Days****> (calculated using* **computeExperience( )**).
- Any input not following the mentioned Input Format should display "INVALID"

TEST CASES

INPUTS	***OUTPUTS***
S Amith b190142cs 9 9	INVALID (as it contains less number of attributes for student)
S Nima b180442ec 10 10 10 10 T Raghu 112 01-07-2008 MTech AssistantProfessor	Name:Nima RollNumber:b180442ec CGPA:10 Name:Raghu EmpID:112 Qualification:MTech Designation:AssistantProfessor Experience:4837 Days

Q3. Write a java program to implement an interface **Queue** which defines the basic operations needed to implement the queue data structure for integers**.** The **Queue** interface contains the following methods:-

- **enqueue(int item):-** Adds an element to the end of the queue.
- **dequeue( ):-** Removes and returns an element from the front of the queue.
- **isEmpty( ):-** Returns true if the queue is empty.
- **isFull ( ):-** Returns true if the queue is full.
- **size( ):-** Returns the number of elements in the queue.
- **display( ):-** Displays the elements in the queue.

Create two classes **ArrayQueue** and **ListQueue** where class **ArrayQueue** is an array-based implementation of the **Queue** interface, and the class **ListQueue** is the linked-list implementation of the **Queue** interface.
Create a test class **TestQueue** to test the operations of **ArrayQueue** and **ListQueue.**

NOTE: *Do not use any inbuilt functions.*

Input format

First line contains **A <size of the array>** or **L**
A <size of array> - Use class **ArrayQueue** for the implementation of the queue
L - Use class **ListQueue** for the implementation of the queue

In the subsequent lines, you can use one of the options below:

- ***E*** to insert an element at the end of the queue (format:- **E** <value>)
- ***D*** to remove and print an element from the front of the queue. (format:- **D**)
- ***P*** to print the elements in the queue (format:- **P**)
- ***S*** to print the number of elements in the queue (format:- **S**)
- **Q** to exit the program

Output format

- If the first line of the input contains **A <size of array**> , and if the size specified is not a positive integer (less than zero), print "***InvalidSIZE***" and terminate the program.
- If the queue is empty and
  - if the input option is **D** or **P** or **S** , then print "***EmptyQueue***"
- If the first line of input contains character **A** , and if the queue is full;
  - if the input option is **E**, then print "***QueueFull***"
- If the input option is **D**, then print the value of the element removed.
- If the input option is **P**, then print the elements in the queue separated by space.
- If the input option is **S**, then print the number of the elements in the queue.
- If the input option is **Q**, then print "***End***", and terminate the program.

Constraints
The elements in the queue are positive integers.

TEST CASES

INPUTS	**OUTPUTS**
A 4 E 10 E 20 E 30 E 40 E 50 P	QueueFull 10 20 30 40 4 10 20 30 40

S D D D D D Q	EmptyQueue End
L P E 5 E 4 E 6 D P S Q	EmptyQueue 5 4 6 2 End
A -10	InvalidSIZE