**PSG COLLEGE OF TECHNOLOGY, COIMBATORE**

**DEPARTMENT OF APPLIED MATHEMATICS AND COMPUTATIONAL SCIENCES**

# 20XW57 – JAVA PROGRAMMING LAB

**PROBLEM SHEET I – Basics)**

Class : M.Sc (SS) Semester : 5

**20pw01-Abishek.A**

# 1)Calculating the Energy Produced by a Wind Turbine

Suppose you wanted to harness to power of wind to generate electricity. One question you might want to ask is: what is the amount of power my wind turbine can produce? To answer this question, you will need at least the following information:

* The average wind speed (in m/s)
* The operating efficiency of your wind turbine (in %)
* The radius of the blades on your wind turbine (in meters)

To compute the maximum power output for your wind turbine, you would need the following math formulas:

* *A= πr2,* which represents the cross-sectional area of a circle (π = 3.14159265).
* *Pmax = 0.5ρAv3*, which calculates the **maximum** available power given the wind speed in m/s (*ν*), cross-sectional area of the blades in m2 (*A*), and the density of the air (*ρ* *=* 1.2 kg/m3).

Once you have the *maximum* available power, computing the **actual** amount of power (not the maximum) produced by the wind turbine is a matter of determining the amount of power based on the operating efficiency.

Create a Java class called WindPower, and save the file as WindPower.java in your Java development environment. This program will ultimately compute the amount of power produced by a wind turbine.

Input /Output

Enter the Average Wind Speed (in m/s): 12.29

Enter the Operating Efficiency [0...1]: 0.10

Enter the Blade Radius (in meters): 3.0

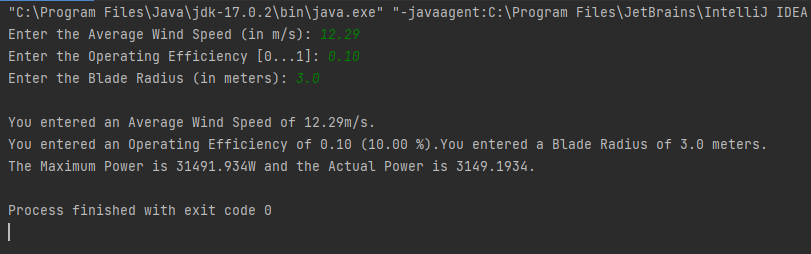
You entered an Average Wind Speed of 12.29 m/s.

You entered an Operating Efficiency of 0.10 (10.00%).

You entered a Blade Radius of 3.00 meters.

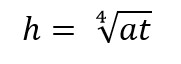
The Maximum Power is 31491.93 W and the Actual Power is 3149.19 W.

import java.util.Scanner;  
import java.lang.Math;  
  
public class WindPower {  
 public static void main(String[] args) {  
 Scanner obj = new Scanner(System.*in*);  
 System.*out*.print("Enter the Average Wind Speed (in m/s): ");  
 float v = obj.nextFloat();  
 System.*out*.print("Enter the Operating Efficiency [0...1]:");  
 float f = obj.nextFloat();  
 System.*out*.print("Enter the Blade Radius (in meters): ");  
 float r = obj.nextFloat();  
 final float pi = 3.14159265f;  
 float A = pi\*(float)Math.*pow*(r,2);  
 float d = 1.2f;  
 float P = 0.5f\*d\*A\*(float)Math.*pow*(v,3);  
 System.*out*.println("\nYou entered an Average Wind Speed of "+v+ "m/s.");  
 System.*out*.printf("You entered an Operating Efficiency of %.2f (%.2f",f,f\*100);  
 System.*out*.print(" %).");  
 System.*out*.println("You entered a Blade Radius of "+r+" meters.");  
 System.*out*.println("The Maximum Power is "+P+"W and the Actual Power is "+P\*f+".");  
 }  
}



# 2)Trees and Growth Patterns

As we plant trees strategically, we often want to know how tall a tree can be expected to grow. While there are environmental factors that impact tree growth patterns, we can approximate the height of tree if we know certain facts. As trees grow, they add approximately the same amount of mass each year. If we know this amount, we can compute the expected height of a tree over time. The height of a tree (*h*) can be approximated based on its age using the following formula:



where *a* = the amount of mass added to a tree each year and *t* is the age in years.

Input /Output

Enter the Amount of Mass: 256.0

Enter the Age in Years: 81

You entered a mass of 256.00 and an age of 81 years.

At age 81 years, the approximate height of the tree is 12.00 meters.

At age 82 years, the approximate height of the tree is 12.04 meters.

At age 83 years, the approximate height of the tree is 12.07 meters.

At age 84 years, the approximate height of the tree is 12.11 meters.

At age 85 years, the approximate height of the tree is 12.15 meters.

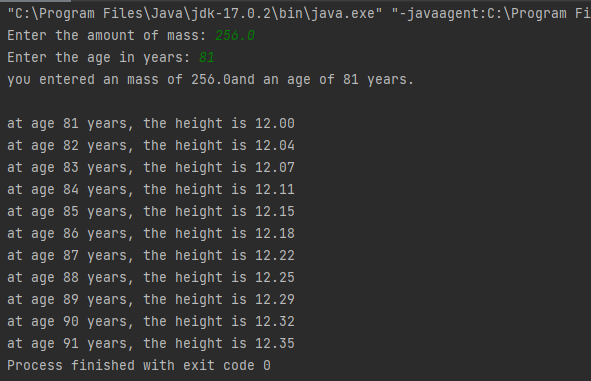
At age 86 years, the approximate height of the tree is 12.18 meters.

At age 87 years, the approximate height of the tree is 12.22 meters.

At age 88 years, the approximate height of the tree is 12.25 meters.

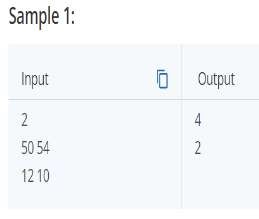
At age 89 years, the approximate height of the tree is 12.29 meters. At age 90 years, the approximate height of the tree is 12.32 meters.

import java.util.Scanner;  
public class Growth {  
  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.*in*);  
 System.*out*.print("Enter the amount of mass: ");  
 float mass = input.nextFloat();  
 System.*out*.print("Enter the age in years: ");  
 int age = input.nextInt();  
 System.*out*.print("you entered an mass of " + mass);  
 System.*out*.print("and an age of " + age +" years.\n");  
 for (int i = age; i <= age + 10; i++) {  
 float j = mass \* i;  
 float h = (float) Math.*pow*(j, 0.25);  
 System.*out*.printf("\nat age %d years, the height is %.2f",i,h);  
 }  
 }  
}

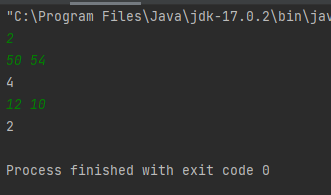


**3) Problem**

Chef is watching TV. The current volume of the TV is X*X*. Pressing the volume up button of the TV remote increases the volume by 11 while pressing the volume down button decreases the volume by 11. Chef wants to change the volume from X to *Y*. Find the minimum number of button presses required to do

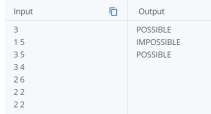


import java.util.\*;  
public class TV {  
  
 public static void main(String[] args) {  
  
 Scanner scan = new Scanner(System.*in*);  
 int n = scan.nextInt();  
 for (int i = 0; i < n; i++) {  
 int a = scan.nextInt();  
 int b = scan.nextInt();  
 int p = a - b;  
 System.*out*.println(Math.*abs*(p));  
 }  
  
 }  
  
}

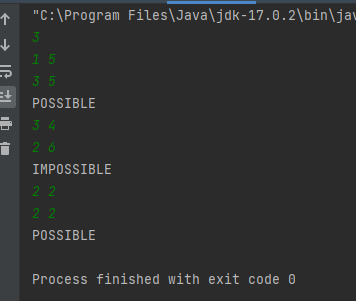


# 4)Problem

Chef is watching a football match. The current score is A:B*A*:*B*, that is, team 11 has scored A*A* goals and team 22 has scored B*B* goals. Chef wonders if it is possible for the score to become C:D*C*:*D* at a later point in the game (i.e. team 11 has scored C*C* goals and team 22 has scored D*D* goals). Can you help Chef by answering his question?



import java.util.\*;  
  
public class Shoes {  
 public static void main(String[] args) {  
  
 Scanner scan = new Scanner(System.*in*);  
  
 int n = scan.nextInt();  
  
 for (int i = 0; i < n; i++) {  
  
 int a = scan.nextInt();  
  
 int b = scan.nextInt();  
  
 int c = scan.nextInt();  
  
 int d = scan.nextInt();  
  
 if ((c<a) || (d<b)) {  
  
 System.*out*.println("IMPOSSIBLE");  
  
 }  
  
 else{  
  
 System.*out*.println("POSSIBLE");  
  
 }  
 }  
 }}



# Problem

Chef has N friends. Chef promised that he would gift a pair of shoes (consisting of one left shoe and one right shoe) to each of his N friends. Chef was about to go to the marketplace to buy shoes, but he suddenly remembers that he already had M left shoes.

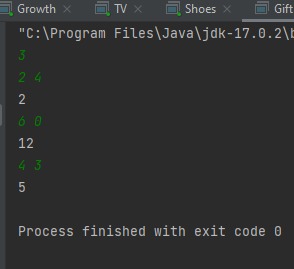
What is the minimum number of extra shoes that Chef will have to buy to ensure that he is able to gift a pair of shoes to each of his N friends?

For example, if N = 2, M = 4, then Chef already has 44 left shoes, so he must buy 22 extra right shoes to form 22 pairs of shoes.

Therefore Chef must buy at least 22 extra shoes to **ensure** that he is able to get N = 2 pairs of shoes.



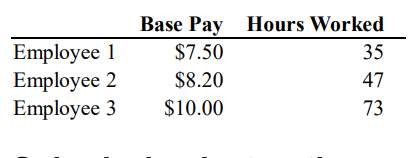
import java.util.Scanner;  
public class Gift {  
 static int buyShoes(int n,int m){  
 if(m>n){  
 return m-n;  
 }  
 return n-m+n;  
 }  
 public static void main (String [] args) {  
 Scanner obj = new Scanner(System.*in*);  
 int t = obj.nextInt();  
 for (int i=0; i<t; i++){  
 int n = obj.nextInt();  
 int m = obj.nextInt();  
 System.*out*.println(*buyShoes*(n,m));  
 }  
 }  
}

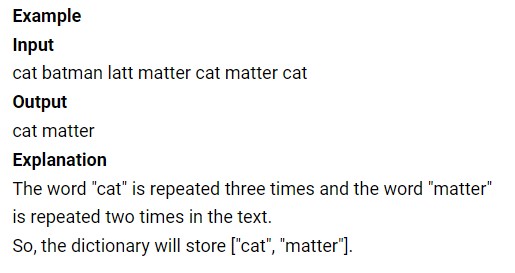
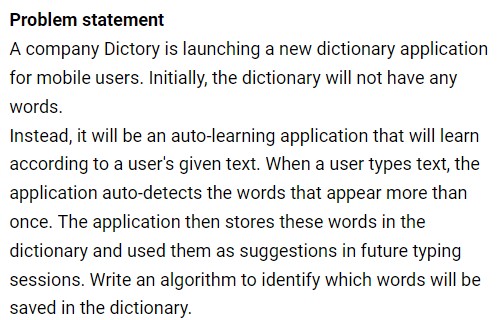
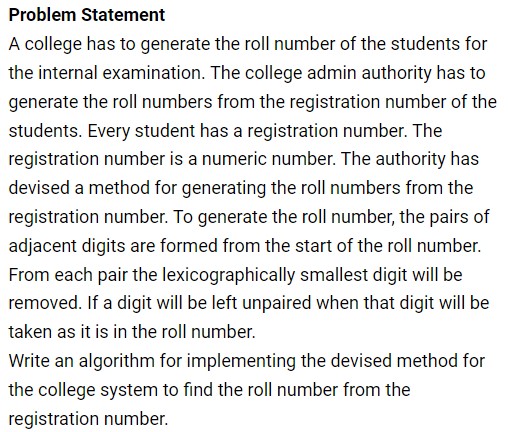
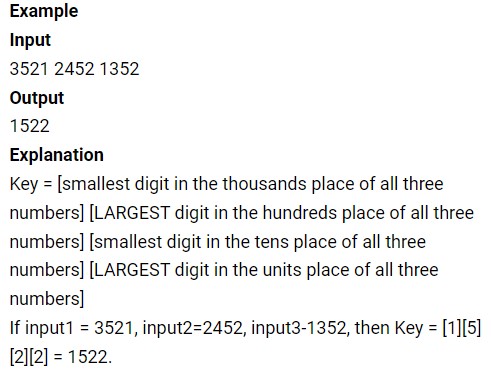
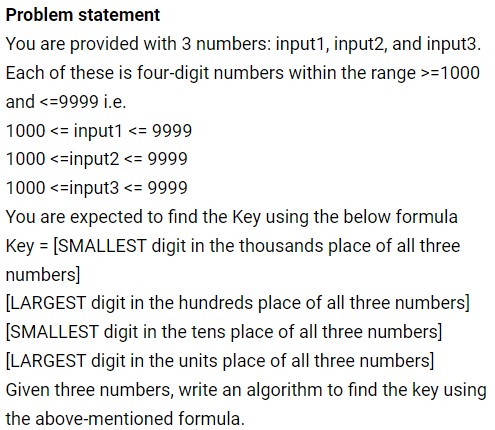
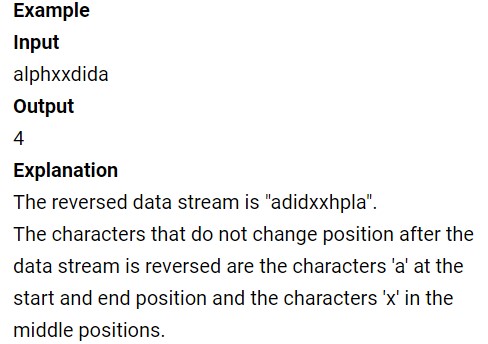
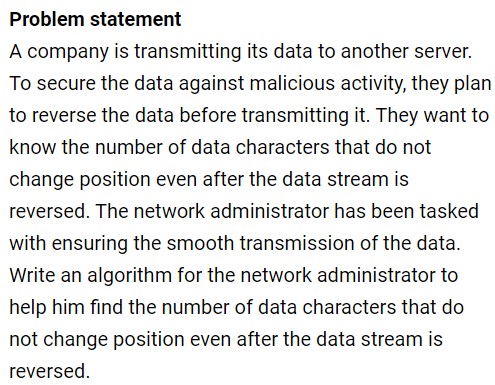


6)Foo Corporation needs a program to calculate how much to pay their hourly employees. The US Department of Labor requires that employees get paid time and a half for any hours over 40 that they work in a single week. For example, if an employee works 45 hours, they get 5 hours of overtime, at 1.5 times their base pay. The State of Massachusetts requires that hourly employees be paid at least $8.00 an hour. Foo Corp requires that an employee not work more than 60 hours in a week.

* An employee gets paid (hours worked) × (base pay), for each hour up to 40 hours.
* For every hour over 40, they get overtime = (base pay) × 1.5.
* The base pay must not be less than the minimum wage ($8.00 an hour). If it is, print an error.
* If the number of hours is greater than 60, print an error message.

Create a new class called FooCorporation. Write a method that takes the base pay and hours worked as parameters, and prints the total pay or an error. Write a main method that calls this method for each of these employees:





**Jumping Champion**

An integer N is called the jumping champion if it is the most frequently occurring difference between consecutive primes.

For example, consider the consecutive primes 2 3 5 7 11.The differences between primes are 1 2 2 4. Therefore, for this set of primes, the jumping champion is 2 (occurring two times).

Write a program that calculates the jumping champion of all the primes numbers that are defined between a lower(L) and an upper(U) bound.

The upper and lower bound are considered themselves to be inside the limit.

[**Note**: L and U (0 ≤ L ≤ U ≤ 1000000)]

**Input and Output Format:**

Refer sample input and output for formatting specifications.

**[All text in bold corresponds to input and the rest corresponds to output.]**

**Sample Input and Output 1:**

Enter the lower bound and upper bound values

**2 11**

The jumping champion is 2

**Sample Input and Output 2:**

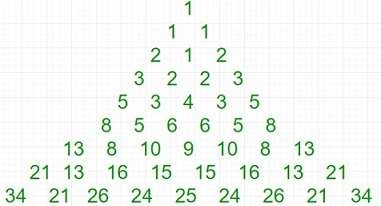
Enter the lower bound and upper bound values

**2 5**

No jumping champion

# Hosoya’s Triangle

The Fibonnaci triangle or Hosoya’s triangle is a triangular arrangement of numbers based on Fibonacci numbers. Each number is the sum of two numbers above in either the left diagonal or the right diagonal. The first few rows are:



The numbers in this triangle follow the recurrence relations:

H(0,0) = H(1,0) = H(1,1) = H(2,1) = 1

and,

H(n,j) = H(n-1,j) + H(n-2, j)

= H(n-1,j-1) + H(n-2, j-2)

Relation to Fibonacci numbers:

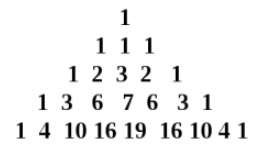
The entries in the triangle satisfy the identity

H(n,i) = F(i+1) x F(n-i+1)

Thus, the two outermost diagonals are the Fibonacci numbers, while the numbers on the middle vertical lines are the squares of the Fibonacci numbers. All the other numbers in the triangle are the product of two distinct Fibonacci numbers greater than 1. The row sums are the first convolved Fibonnaci numbers.

# Trinomial Triangle

The trinomial triangle is a variation of Pascal’s triangle. The difference between the two is that an entry in the trinomial triangle is the sum of the three (rather than the two in Pasacal’s triangle) entries above it :



The k-th entry of the n-th row is denoted by :



Rows are counted starting from 0. The entries of the n-th row are indexed starting with -n from the left, and the middle entry has index 0. The symmetry of the entries of a row about the middle entry is expressed by the relationship



**Input Format:**

The input consists of a integer value which represents the N value.

**Output format:**

Output consists of a trinomial triangle of N lines.

**Sample Input 1:**

4

**Sample Output 1:**

1

1 1 1

1 2 3 2 1

1 3 6 7 6 3 1

**Sample Input 2:**

5

**Sample Output 2:**

1

1 1 1

1 2 3 2 1

1 3 6 7 6 3 1

1 4 10 16 19 16 10 4 1