Question 1

Write a program that checks the size of an IP packet sent over a WiFi interface and decides if the size is valid for both IP and WiFi packet size rules or not. The maximum IP packet size is 65536 bytes and the maximum WiFi packet size is 7981 bytes. The program will take **one integer value (**packet_size) in bytes.

If the packet_size is bigger than the IP packet size limit, **the program will print, "The packet is too big**." Otherwise, the program will check to how many WiFi packets the IP packet can fit. If packet_size is bigger than the WiFi packet size limit, the program will fragment (i.e., divide) the IP packet into smaller WiFi packets where each WiFi packet has a maximum value of 7981 bytes. As the output, **the program will print out the number of WiFi packets the IP packet has divided into and the size of the last packet,both as integer values**.

Example: In the Input/Output #2 below, the IP packet is bigger than two times the WiFi packet size, so the IP packet is divided into 3 WiFi packets where the first two WiFi packets have a size of 7981 and the last WiFi packet has a size of 78.

NOTE: If the IP packet size is less than the WiFi packet size, the program will just print out "1" and the IP packet size since in this case there is no need for fragmentation.

Input	80000	16040	718
Output	The packet is too big	3 78	1 718

Question 2

Write a program that reads **5 integer numbers** from the user and returns the **DIFFERENCE** between the 3rd input number and the arithmetic mean of all 5 values **as a double value** (i.e., 3rd number – arithmetic mean).

NOTE: You are expected to use **AT LEAST ONE** array variable to store the values.

Input	1 2 3 4 5	1 2 3 7 8	9 8 7 3 2
Output	0.000000	-1.200000	1.200000

Question 3

Write a program that reads **two integer values**, n and k, calculates the formula below (i.e., F(n)), and prints the result to the screen **as an integer value**.

$$F(n) = \prod_{i=0}^{k} (n-i)!$$

NOTE: The value of k will **ALWAYS** be less than or equal to N (i.e., k<=n)

NOTE: The value of n will be **AT MOST** 16 (i.e., n<= 16)

HINT: Writing a method that calculates the factorial of a given number and using it in the program is highly recommended.

Input	3 3	5 3	10 4
Output	12	34560	1073741824

Question 4

Write a program that calculates the total cost of a travel plan. First, the number of stops in the trip (including the initial city) and the total number of cities will be given both **as integers**. Next, the travel plan will be given as a series of **integer city indexes** in the travel order (i.e., the first city index refers to the start location of the travel and the last city index refers to the last stop in the travel plan). Lastly, the costs of directly travelling between cities will be given in **a 2D double array**.

Based on the costs and the travel plan, your program will calculate the total cost of the travel plan and print it out **as a double value**. Also, a special rule affects the total cost. If the 2nd or the 4th city is visited during the travelling plan. For each of these cities add 2.45 to the total cost (e.g., if only 2nd city is visited add 2.45 to the total cost, if both are visited add 4.9). In case there are mulitple visits to these cities, only add this value once per city.

NOTE: The maximum number of cities is 20, and the maximum number of steps in the travel plan is 10.

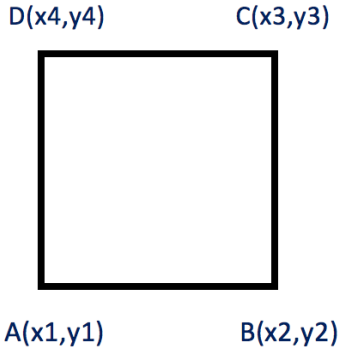
NOTE: You are expected to use **AT LEAST ONE** 2D array variable in this question.

Input	4 4 3 1 2 3 0.0 2.3 4.5 6.7 2.3 0.0 5.9 3.4 4.5 5.9 0.0 11.1 6.7 3.4 11.1 0.0	6 5 1 2 3 4 5 1 0.0 1.2 3.4 5.6 9.8 1.2 0.0 4.6 7.2 3.9 3.4 4.6 0.0 8.5 9.2 5.6 7.2 8.5 0.0 6.1 9.8 3.9 9.2 6.1 0.0	4 4 4 3 4 2 0.0 12.1 4.5 7.7 12.1 0.0 5.4 3.1 4.5 5.4 0.0 8.8 7.7 3.1 8.8 0.0
Output	15.150000	35.100000	25.600000

Question 5

You are given three classes named Point2d, Line, and Program as seen below. Write a class named **Square** so that the Program class works correctly and calculates the circumference and diagonal of a square.

The three given global classes **MUST** be put inside separate Java files and you **MUST NOT CHANGE** anything in them. Inputs of the Program class will be the eight double numbers representing the 2D coordinates of vertices A(x1,y1), B(x2,y2), C(x3,y3) and D(x4,y4) of a square respectively as seen below.



NOTE: In the methods of your Square class, you **MUST** use an object derived from the **Line class** and use its **calcLength()** method to calculate the length of an edge. Calculating the length value **DIRECTLY** inside your Square class is **NOT** allowed.

NOTE: You will **ONLY** upload the "Square.java" file that you've written to the Canvas. It'll be tested/graded with the three other classes given below.

Input	0 0 2 0 2 2 0 2	8 14 12 16 14 12 10 10	8 3 7 8 2 7 3 2
Output	8.000000 2.828427	17.888544 6.324555	20.396078 7.211103

Class #1: Program:

```
import java.util.Scanner;
public class Program {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        double x1 = input.nextDouble();
        double y1 = input.nextDouble();
        double x2 = input.nextDouble();
        double y2 = input.nextDouble();
        double x3 = input.nextDouble();
        double y3 = input.nextDouble();
        double x4 = input.nextDouble();
```

```
        double y4 = input.nextDouble();
        input.close();
        Point2d point1 = new Point2d(x1,y1);
        Point2d point2 = new Point2d(x2,y2);
        Point2d point3 = new Point2d(x3,y3);
        Point2d point4 = new Point2d(x4,y4);
        Square square = new Square(point1,point2,point3,point4);
        System.out.printf("%f",square.calculateCircumference());
        System.out.printf(" %f", square.calculateDiagonal());
    }
}
```

Class #2: Point2d:

```
public class Point2d {
    private double x;
    private double y;
    public Point2d(double x1,double y1) {
        setX(x1);
        setY(y1);
    }
    public double getX() { return x; }
    public void setX(double x_) { x = x_; }
    public double getY() { return y; }
    public void setY(double y_) { y = y_; }
}
```

Class #3: Line:

```
public class Line {
    private Point2d point1;
    private Point2d point2;
    public Line(Point2d p1, Point2d p2) {
        point1 = p1;
        point2 = p2;
```

```
    }
    public double calcLength() {
        double length = Math.sqrt(Math.pow(point1.getX()-point2.getX(),2) + Math.pow(point1.getY() -
point2.getY(),2));
        return length;
    }
}
```