Lab: Objects and Classes

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

You can check your solutions here: https://judge.softuni.bg/Contests/175/Objects-and-Classes-Lab.

I. Using the Built-in .NET Classes

1. Day of Week

You are given a **date** in format **day-month-year**. Calculate and print the **day of week** in **English**.

Examples

Input	Output
18-04-2016	Monday
27-11-1996	Wednesday

Hints

- **Read the date as string** from the Console.
- Use the method **DateTime.ParseExact(string date, format, provider)** to convert the input string to object of type **DateTime**. Use format **"d-M-yyyy"** and **CultureInfo.InvariantCulture**.
  - Alternatively split the input by "**-**" and you will get the day, month and year as numbers. Now you can create **new DateTime(year, month, day)**.
- The newly created **DateTime** object has property **DayOfWeek**.

2. Randomize Words

You are given a **list of words in one line**. **Randomize their order** and print each word at a separate line.

Examples

Input	Output	Comments
Welcome to SoftUni and have fun learning programming	learning Welcome SoftUni and fun programming have to	The order of the words in the output will be different after each program execution.

Hints

- **Split** the input string by (space) and create an **array of words**.
- Create a random number generator – an object **rnd** of type **Random**.
- In a **for-loop exchange each number** at positions 0, 1, … **words.Length-1** by a number at **random position**. To generate a random number in range use **rnd.Next(minValue, maxValue)**. Note that by definition **minValue** is **inclusive**, but **maxValue** is **exclusive**.
- Print each word in the array on new line.

3. Big Factorial

Calculate and print **n!** (n factorial) for very big integer n (e.g. 1000).

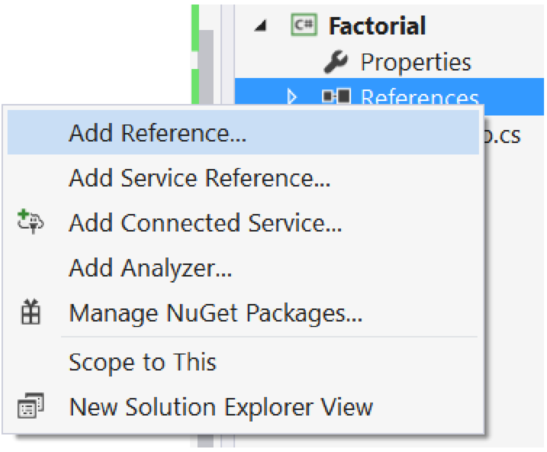
Examples

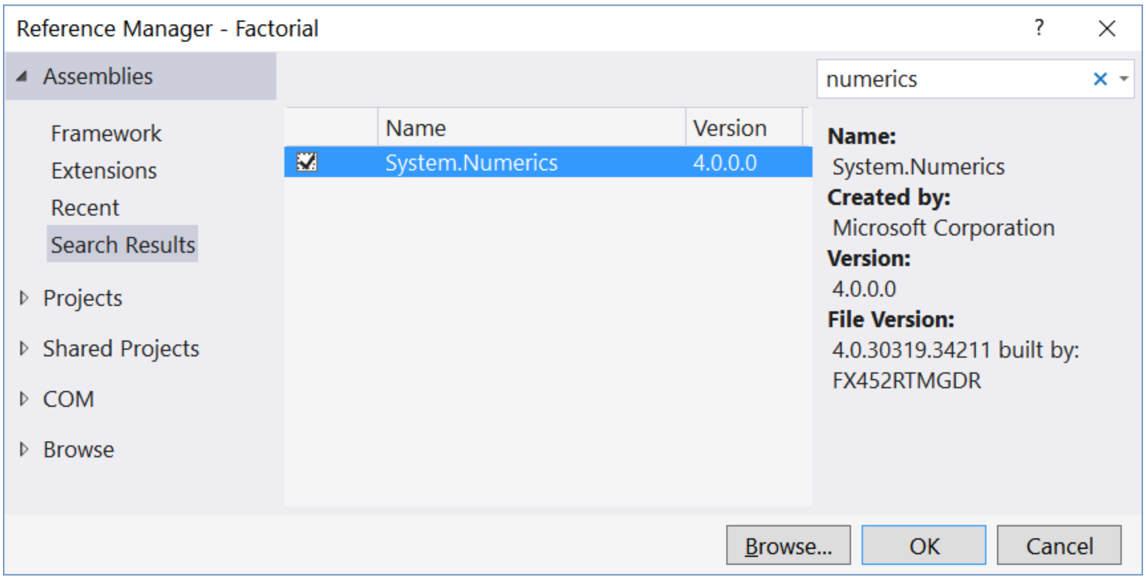
Input	Output
5	120
50	30414093201713378043612608166064768844377641568960512000000000000

Hints

Use the class **BigInteger** from the built-in .NET library **System.Numerics.dll**.

1. Add reference to **System.Numerics.dll**.





☑ System.Numerics 4.0.0.0

Name:
System.Numerics
Created by:
Microsoft Corporation
Version:
4.0.0.0
File Version:
4.0.30319.34211 built by: FX452RTMGDR

2. Import the namespace "**System.Numerics**":

```
using System.Numerics;
```

3. Use the type **BigInteger** instead of **long** or **decimal** to keep the factorial value:

```
BigInteger factorial = 1;
for (int i = 1; i <= n; i++)
    // TODO
```

II. Defining Simple Classes

4. Distance Between Points

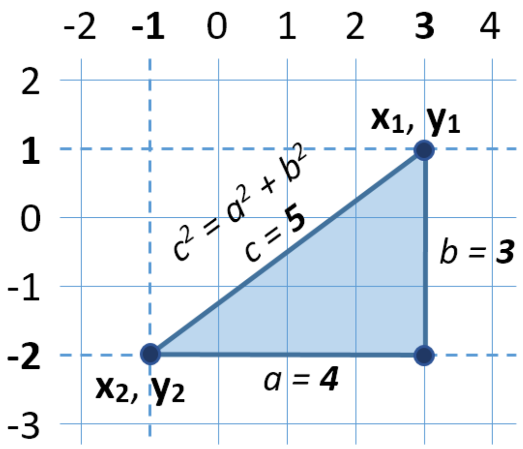
Write a method to calculate the distance between two points **p_1** {**x_1**, **y_1**} and **p_2** {**x_2**, **y_2**}. Write a program to read **two points** (given as two integers) and print the **Euclidean distance** between them.

Examples

Input	Output
3 4 6 8	5.000
3 4 5 4	2.000
8 -2 -1 5	11.402

Hints

- Create a **class `Point`** holding properties **X** and **Y**.
- Write a method **`CalcDistance(Point p1, Point p2)`** that returns the distance between the given points – a **`double`** number.
- Use [this formula]() to calculate the distance between two points. How it works?
  - Let's have two points **p_1** {**x_1**, **y_1**} and **p_2** {**x_2**, **y_2**}
  - Draw a right-angled triangle
  - Side **a** = $|x_1 - x_2|$
  - Side **b** = $|y_1 - y_2|$
  - Distance == side **c** (hypotenuse)
  - $c^2 = a^2 + b^2$ (Pythagorean theorem)
  - Distance = **c** = $\sqrt{a^2 + b^2}$



- You can use [**`Math.Sqrt(number)`**]() method for calculating a square root.

5. Closest Two Points

Write a program to read **n** points and find the **closest two** of them.

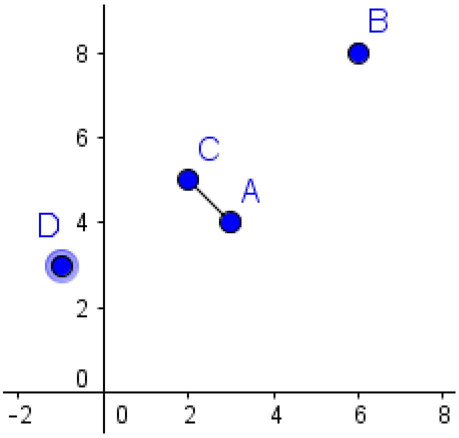
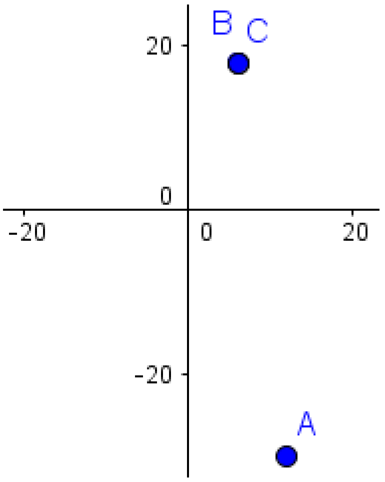
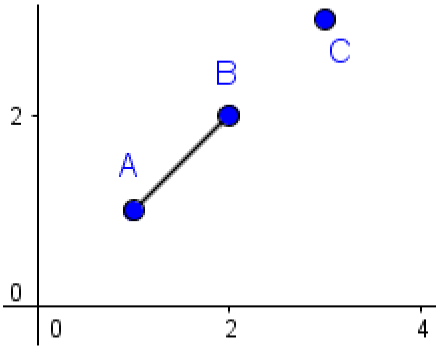
Input

The **input** holds the number of points **n** and **n** lines, each holding a point {**X** and **Y** coordinate}.

Output

- The **output** holds the shortest distance and the closest two points.
- If several pairs of points are equally close, print **the first** of them (from top to bottom).

Examples

Input	Output	Visualization	Comments
4 3 4 6 8 2 5 -1 3	1.414 (3, 4) (2, 5)		The closest two points are **{3, 4}** and **{2, 5}** at distance 1.4142135623731 ≈ **1.414**.
3 12 -30 6 18 6 18	0.000 (6, 18) (6, 18)		Two of the points have the same coordinates **{6, 18}**, so the distance between them is **0**.
3 1 1 2 2 3 3	1.414 (1, 1) (2, 2)		The pairs of points {{1, 1}, {2, 2}} and {{2,2}, {3,3}} stay at the same distance, but the first pair is **{{1, 1}**, **{2, 2}**}. The distance between them is 1.4142135623731 ≈ **1.414**.

Hints

- Use the **class `Point`** you created in the previous task.
- Create an array **`Point[] points`** that will keep all points.
- Create a method **`Point[] FindClosestPoints(Point[] points)`** that will check distance **between every two pairs** from the array of points and returns the two closest points in a new array.
- Print the **closest distance** and the **coordinates** of the two closest points.

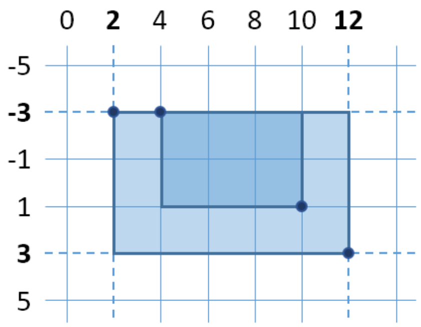
6. Rectangle Position

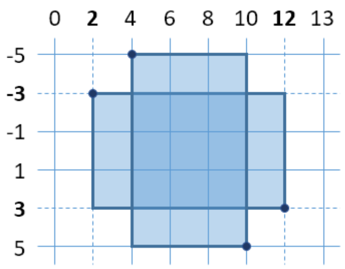
Write a program to **read two rectangles** {left, top, width, height} and print whether the first is inside the second.

The input is given as two lines, each holding a rectangle, described by 4 integers: **left**, **top**, **width** and **height**.

Examples

Input	Output	Visualization	Comments

`4 -3 6 4`
`2 -3 10 6`
	`Inside`	 0
		2
		4
		6
		8
		10
		12
		-5
		-3
		-1
		1
		3
		5
			The first rectangle stays **inside** the second.

`2 -3 10 6`
`4 -5 6 10`
	`Not inside`	 0
		2
		4
		6
		8
		10
		12
		13
		-5
		-3
		-1
		1
		3
		5
			The rectangles intersect, no the first is **not inside** the second.

Hints

- Create a class **`Rectangle`** holding properties **`Top`**, **`Left`**, **`Width`** and **`Height`**.
- Define calculated properties **`Right`** and **`Bottom`**.
- Define a method **`bool IsInside(Rectangle r)`**. A rectangle **`r1`** is inside another rectangle **`r2`** when:
  - **`r1.Left ≥ r2.Left`**
  - **`r1.Right ≤ r2.Right`**
  - **`r1.Top ≤ r2.Top`**
  - **`r1.Bottom ≤ r2.Bottom`**
- Create a method to **read** a **`Rectangle`**.
- Combine all methods into a single program.

7. Sales Report

Write a class **`Sale`** holding the following data: **town**, **product**, **price**, **quantity**. Read a **list of sales** and calculate and print the **total sales by town** as shown in the output. Order **alphabetically** the towns in the output.

Examples

Input	Output	Comments

`5`
`Sofia beer 1.20 160`
`Varna chocolate 2.35 86`
`Sofia coffee 0.40 853`
`Varna apple 0.86 75.44`
`Plovdiv beer 1.10 88`
	`Plovdiv -> 96.80`
	`Sofia -> 533.20`
	`Varna -> 266.98`
		Plovdiv -> 1.10 * 88 = 96.80
		Sofia -> 1.20 * 160 + 0.40 * 853 = 533.20
		Varna -> 2.35 * 86 + 0.86 * 75.44 = 266.98

Hints

- Define the class **`Sale`** holding properties **`Town`**, **`Product`**, **`Price`** and **`Quantity`**.
- Create a method **`ReadSale()`** that reads a sale data line from the console and returns **`Sale`** object. It could split the input line by space and parse the price and quantity.
- To read the input, first read an integer **n**, then **n** times read a sale.
- **Approach I – LINQ**
  - Using **LINQ** select the **distinct town names** from the array of sales and sort them.
  - For **each town** in a loop use a LINQ query to calculate the **total sales** (aggregate the sum of **price** * **quantity** for all sales by the current town).
- **Approach II – Dictionary {town → sales}**
  - Define a dictionary **`SortedDictionary<string, decimal> salesByTown`** to hold the total sales for each town.
  - Pass through all the sales from the input in a loop and for each sale, add its **price** * **quantity** to the **`salesByTown`** for the current **`town`**. If the town is missing in the dictionary, first create it.
  - Finally print the dictionary.
- The second approach is faster, because it scans the array of sales only once.