# Problem 1 – Nacepin

“**Nacepin**” is sold in three different stores – US, UK and China. In every store the product price is in local currency for different packages.

Your task is to write a program, which finds the lowest price in leva per kilogram. You’ll be given **the price for the product in every store** in its local currency and **the kilograms of for every package**. The currency rates are as follows:

* 1 BGN = 0.58 USD
* 1 BGN = 0.41 GBP
* 0.27 BGN = 1 CNY

Your output should print **which store has the lowest leva per kilogram** and **the difference between the lowest and highest prices per kilogram**.

## Input

The input data should be read from the console. You will always receive exactly six lines of input.

* On the first line –the **price** from the **US** online store.
* On the second line – box **weight** in the **US** online store in kilograms**.**
* On the third line – the **price** from the **UK** online store.
* On the fourth line – box **weight** in the **UK** online store in kilograms.
* On the fifth line – the **price** from the **Chinese** online store.
* On the sixth line – box **weight** in the **Chinese** online store in kilograms.

The input data will always be valid and in the format described. There is no need to check it explicitly.

## Output

The output data should be printed on the console.

At the **first line** print the **lowest price per kilogram** in the following format:

“{**country**} store. {**price**} lv/kg”

It is guaranteed that **only one lowest price** will exist.

At the **second line** print the **difference** between the **highest** and the **lowest prices** **per kilogram** in the following format:

“Difference {**difference**} lv/kg”

Moneyshould be formatted with **two digits after the decimal point**.

## Constraints

* The **prices** of Nacepin **in different stores** will be floating point numbers in the range [0 – 7.9 x 1028].
* The **box weight** will be integers in the range [0 - 232].

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10  1  9  2  8  3 | Chinese store. 0.72 lv/kg  Difference 16.52 lv/kg | US store 🡪 $10 per 1 kg. $10 / 0.58 = 17.24 lv. for 1 kg.  17.24 / 1kg = **17.24** lv/kg  UK store 🡪 £9 per 2 kg. £9 / 0.41 = 21.95 lv. for 2 kg.  21.95 / 2kg = **10.98** lv/kg  Chinese store – ¥8 for 3 kg. ¥8 \* 0.27 = 2.16 lv. for 3 kg.  2.16 / 3kg = **0.72** lv/kg  Lowest = 0.72 lv/kg (from the Chinese store)  Highest = 17.24 lv/kg (from the US store)  Saved = 17.24 - 0.72 = 16.52 lv/kg |
| **Input** | **Output** | **Comments** |
| 10  1  10  1  10  1 | Chinese store. 2.70 lv/kg  Difference 21.69 lv/kg | US store 🡪 $10 per 1 kg. $10 / 0.58 = 17.24 lv. for 1 kg.  17.24 / 1kg = **17.24** lv/kg  UK store 🡪 £10 per 1 kg. £10 / 0.41 = 24.39 lv. for 1 kg.  24.39 / 1kg = **24.39** lv/kg  Chinese store 🡪 ¥10 for 1 kg. ¥10 \* 0.27 = 2.70 lv. for 1 kg.  2.70 / 1kg = **2.70** lv/kg  Lowest = 2.70 lv/kg (from the Chinese store)  Highest = 24.39 lv/kg (from the UK store)  Saved = 24.39 - 2.70 = 21.69 lv/kg |

**Problem 2 – Hogwarts Sorting**

Do you remember that sorting hat from Harry Potter? It turns out that the hat needed some rest, so it took a vacation (here is where you imagine a hat lying on the beach \*WHAT?!\*). So Hogwarts had to come up with something which will help them sort all newcomers while the hat is absent. They decided to keep it as simple as possible – the algorithm is such, that it **takes all the letters from the student’s name and sums their ASCII codes**. Then the **sum is divided by 4** and depending on the **reminder**, the student is sorted in one of the houses – **Gryffindor** (if **reminder** = **0**), **Slytherin** (if **1**)**, Ravenclaw** (if **2**) and **Hufflepuff** (if **3**). What is more, for greater efficiency, they plan to introduce a system of faculty numbers, which are composed by **the sum**, followed by the **initials of the names** of the student. Great algorithm, but they needed someone to **write a program** which sorts the students with the given algorithm, so they asked you. You will be **given the number of newcomers** and then **their names, each on a separate line. You just have to do the magic!**

**Input**

* On the first line you will receive the number **N** – the number of newcomers.
* On the next **N** lines you will receive valid names of the students

**Output**

* For each student you have to print on a separate line the house in which he is sorted and his faculty number, separated by a space
* Then on the next 4 separate lines you print the name of each house and the count of students sorted there.  
  ***Example: “Gryffindor: 5”***

**Constraints**

* **N** is an integer in range [1 … 30]
* The **names of the student** will be valid strings, composed of first and last name.
* Allowed memory: 16 MB
* Allowed time: 0.1s

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  Harry Potter  Draco Malfoy  Hermione Granger | Gryffindor 1156HP  Slytherin 1105DM  Slytherin 1533HG  Gryffindor: 1  Slytherin: 2  Ravenclaw: 0  Hufflepuff: 0 | **72** (H) + **97** (a) + **114** (r) + **114** (r) + **121** (y) + **80** (P) + **111** (o) + **116** (t) + **116** (t) + **101** (e) + **114** (r) = **1156**.  1156 / 4 = 289(0) **🡪** Gryffindor  Faculty number: **1156HP** |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Ivan Petrov  Pesho Programist  Leliya Vanya  Dedo Gosho | Ravenclaw 1038IP  Hufflepuff 1575PP  Hufflepuff 1119LV  Gryffindor 892DG  Gryffindor: 1  Slytherin: 0  Ravenclaw: 1  Hufflepuff: 2 |

# Problem 3 – Dress Pattern

One of Emma’s favorite things is to sew dresses. Her friends like the dresses the makes so much that they want her to make some for them. She needs to optimize the process and would like you to make a good resizable dress pattern for her. Use **dot ‘.’ f**or the outlines of the dress and the lowercase letter **‘o’** for the belt. Use underscore **‘\_’** for the rest.

The **total width** with the sleeves is **12\*n + 2**. The **horizontal width** of the sleeve to the armpit is **3\*n**.

See the examples for more clarity.

**Input**

The input should be read from the console. It will consist of a single line, holding a positive number.

**Output**

The output should be the dress resized based on the input number;

**Constraints**

* The input will be a valid integer, in the range [1 … 100].
* Allowed working time for your program: 0.25 seconds.
* Allowed memory: 16MB.

**Examples**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 2 | \_\_\_\_\_\_\_\_.\_\_\_\_\_\_\_\_.\_\_\_\_\_\_\_\_  \_\_\_\_\_\_.\*\*.\_\_\_\_\_\_.\*\*.\_\_\_\_\_\_  \_\_\_\_.\*\*\*\*\*.\_\_\_\_.\*\*\*\*\*.\_\_\_\_  \_\_.\*\*\*\*\*\*\*\*.\_\_.\*\*\*\*\*\*\*\*.\_\_  .\*\*\*\*\*\*\*\*\*\*\*..\*\*\*\*\*\*\*\*\*\*\*.  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  ......\*\*\*\*\*\*\*\*\*\*\*\*\*\*......  \_\_\_\_\_\_oooooooooooooo\_\_\_\_\_\_  \_\_\_\_\_\_oooooooooooooo\_\_\_\_\_\_  \_\_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_\_  \_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_  \_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_  \_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_  \_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_  \_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_  .......................... |  | 3 | \_\_\_\_\_\_\_\_\_\_\_\_.\_\_\_\_\_\_\_\_\_\_\_\_.\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_.\*\*.\_\_\_\_\_\_\_\_\_\_.\*\*.\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_.\*\*\*\*\*.\_\_\_\_\_\_\_\_.\*\*\*\*\*.\_\_\_\_\_\_\_\_  \_\_\_\_\_\_.\*\*\*\*\*\*\*\*.\_\_\_\_\_\_.\*\*\*\*\*\*\*\*.\_\_\_\_\_\_  \_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_  \_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*..\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  .\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.  .........\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.........  \_\_\_\_\_\_\_\_\_oooooooooooooooooooo\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_oooooooooooooooooooo\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_oooooooooooooooooooo\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_\_\_  \_\_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_\_  \_\_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_\_  \_\_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_\_  \_\_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_\_  \_\_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_\_  \_.\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*.\_  ...................................... |

# Problem 4 – Vehicle Park

You are manager on a vehicle park. Your job is to sell cars and give reports to the accounting. You will be given all vehicles that are available for selling in format like the example below:

**c2 c4 v10 v20 b50**

Each car is described by **vehicle type** (single character ‘**b**’, ‘**c**’ or ‘**v**’) and **amount of seats** in the vehicle (natural number). For example, **“c4”** means **car with 4 seats**, “**b50**” means **bus with 50 seats** and “**v10**” means **van with 10 seats**.

You need to **process** a sequence of **incoming requests**. Each request holds **type of vehicle** and **amount of seats** in the following format:

**Car with 4 seats**

**Bus with 20 seats**

**…**

If you have **vehicle** that **matches** the description of the **desired vehicle**, you should **sell it**, otherwise print “**No**”. The **price** is calculated as а **product** of the **character ASCII code** andthe **number of seats**. For example, the **price** for “**c4**” (**car with 4 seats**), will be calculated as **99(‘c’) \* 4 = 396**. If there are **2 or more matching vehicles** you should **sell the leftmost** one.

After you **run out of customers**, you need to **print the vehicles** that you **didn’t sell** and the **count of sold vehicles**.

## Input

The input data should be read from the console.

* On the **first input line** you will receive **all vehicles** in the park, separated with **single whitespace**.
* On the nextlines you will receive **requests for vehicles** in the following format:

“**{Vehicle Type} with {Amount of seats} seats**”

until you receive **“End of customers!”**

The input data will always be valid and in the format described. There is no need to check it explicitly.

## Output

The output should consist of:

* For **each vehicle request** you either need to **print**:
  + “Yes, sold for {price}$” – if the **wanted vehicle** is **available** in the park.
  + “No” – if there is no such vehicle in the vehicle park.
* After you **stop receiving request**, you need to **print** **two** **lines**:
  + On the first line you need to print the remaining vehicles in the format:

“Vehicles left: x1, x2, x3…”

* + On the second line you need to print the total amount of vehicles sold in the following format: “Vehicles sold: x1, x2, x3…”

## Constraints

* The **amount of vehicles** will be in range [**0 – 10,000**].
* The **amount of request** for vehicles will be in range [**0 – 10,000**].
* The **amount of seats** for **each vehicle** will be in range [**1 – 10,000**].
* The **vehicle type** can only be one of the following **Car – c**; **Bus – b**; **Van – v;**
* Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| c2 c4 v10 v20 b50  Car with 4 seats  Bus with 20 seats  Bus with 33 seats  Van with 20 seats  Bus with 50 seats  End of customers! | Yes, sold for 396$  No  No  Yes, sold for 2360$  Yes, sold for 4900$  Vehicles left: c2, v10  Vehicles sold: 3 | c4 -> 99(‘c’) \* 4 = 396$  v20 -> 118(‘v’) \* 20 = 2360$  b50 -> 98(‘b’) \* 50 = 4900$ |
| **Input** | **Output** | |
| c2 v1 b2 v2 c20 b150 v1  Van with 50 seats  Van with 1 seats  Bus with 1000 seats  End of customers! | No  Yes, sold for 118$  No  Vehicles left: c2, b2, v2, c20, b150, v1  Vehicles sold: 1 | |

**Problem 5 – Blur Filter**

Bojo is a bad photo editor, but he wants to do some amazing pictures for his Facebook page. He can’t do it alone, so he needs your help. For each picture you will be given a **matrix** with pixels. Each pixel has **weight**. The **blur filter** is applied to a certain cell (pixel) and **all cells around** it. **The blur** has **amount**, which needs to be **added to the weight of the pixel** that it blurs. Print the matrix after the blur applied as output.

**Example:** on the picture on the left **apply blur** with amount **2** over the **pixel** at position **[2, 2]**.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 🡪 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 5 | 8 | 9 | 10 |
| 9 | 10 | 11 | -12 | 9 | 12 | 13 | -10 |
| -13 | 14 | 15 | 16 | -13 | 16 | 17 | 18 |

### Input

The input data should be read from the console.

* The **first line** holds the **blur amount**.
* The **second line** holds the number of rows **r** andcolumns **c** separated by aspace.
* The **next r lines** hold the matrix numbers. Each lines holds **c** integers, separated by space.
* The **last line** holds the **coordinates** **of the blur** – **row** and **column**, separated by space.

The input data will always be valid and in the format described. There is no need to check it explicitly.

### Output

The output should consist of the matrix after the blur filter is applied.

### Constraints

* The **blur amount** will be an integer number in the range [-2,147,483,648...2,147,483,647].
* The **pixel** **weight** will be an integer number in range [-2,147,483,648...2,147,483,647].
* The number of **rows** and **columns** will be an integer number in the range [1...20].

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 9  3 3  1 1 1  1 1 1  1 1 1  1 1 | 10 10 10  10 10 10  10 10 10 | **Blur amount = 9**  Target = **[1, 1]**  **[0, 0] = 1+9; [0, 1] = 1+9;** **[0, 2] = 1+9;**  **[1, 0] = 1+9; [1, 1] = 1+9;** **[1, 2] = 1+9;**  **[2, 0] = 1+9; [2, 1] = 1+9;** **[2, 2] = 1+9;** |
| **Input** | **Output** | |
| 3  3 4  0 -5 4 20  0 20 4 -5  20 4 -5 0  1 2 | 0 -2 7 23  0 23 7 -2  20 7 -2 3 | |