# Problem 1. Numbers

Write a program to **read a sequence of integers** and find and print the **top 5** numbers that are **greater than the average** value in the sequence, sorted in descending order.

## Input

Read from the console a single line holding space separated number.

## Output

Print the above described numbers on a single line, space separated. If **less than 5 numbers** hold the above mentioned property, print less than 5 numbers. Print “**No**” if no numbers hold the above property.

## Constrains

All input numbers are integers in range [-1 000 000 … 1 000 000]. The count of numbers is in range [1…10 000].

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10 20 30 40 50 | 50 40 | Average number = 30.  Numbers greater than 30 are: {40, 50}.  The top 5 numbers among them in descending order are: {50, 40}.  Note that we have only 2 numbers, so all of them are included in the top 5. |
| 5 2 3 4 -10 30 40 50 20 50 60 60 51 | 60 60 51 50 50 | Average number = 28.08.  Numbers greater than 20.078 are: {30, 40, 50, 50, 60, 60, 51}.  The top 5 numbers among them in descending order are: {60, 60, 51, 50, 50}. |
| 1 | No | Average number = 1.  There are no numbers, greater than 1. |
| -1 -2 -3 -4 -5 -6 | -1 -2 -3 | Average number = -3.5.  Numbers greater than -3.5 are: {-1, -2, -3}.  The top 5 numbers among them in descending order are: {-1, -2, -3}. |

# Problem 2. SoftUni Coffee Orders

We are placing **N** orders at a time. You need to calculate the price after the discount based on the following formula:

((daysInMonth \* capsulesCount) \* pricePerCapsule)

\***Hint** – The DateTime class may come in handy to calculate the days in month.

### Input / Constraints

* On the first line you will receive integer **N** – the count of orders the shop will receive.
* For each order you will receive the following information:
  + Price per capsule - **floating-point number in range [0…79,228,162,514,264,337,593,543,950,335]**.
  + Order date - in the following format {d/M/yyyy}, e.g. 25/11/2016, 7/03/2016, 1/1/2020.
  + Capsules count - **integer in range [0…2,147,483,647]**.

The input will be in the described format, there is no need to check it explicitly.

### Output

The output should consist of **N + 1** lines. For each order you must print a single line in the following format:

* “The price for the coffee is: ${**price**}”

On the last line you need to print the total price in the following format:

* “Total: ${totalP**rice**}”

The **price must be rounded** to 2 decimal places.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1  1.53  06/06/2016  8 | The price for the coffee is: $367.20  Total: $367.20 | We are given only 1 order. Then we use the formulas:  **orderPrice** = 30 (days in June 2016) \* 8 \* 1.53 = 367.20 |

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** |  |
| 2  4.99  6/07/2016  3  0.35  03/01/2013  5 | The price for the coffee is: $464.07  The price for the coffee is: $54.25  Total: $518.32 |

# Problem 3. Football Standings

**You will be given** information about **results of football matches**. **Create a standings table by points.** For **win the team gets 3** **points**, **for loss – 0** and **for draw – 1**. Also **find the top 3 teams with most scored goals in descending order**. If **two or more teams** are with **same goals scored** or **same points** **order them by name** in **ascending order**.

The name of each team is **encrypted**. You must **decrypt it before** **proceeding with calculating** statistics. You will be given some **string key** and the **team name will be placed between that key in reversed order.**

**For example:** the **key**: “???”;

**String to decrypt**: “kfle???airagluB???gertIt%%” -> “airagluB” -> “Bulgaria”

Also you should **ignore the letter casing** in the team names. **For example:**

buLgariA = BulGAria = bulGARIA = BULGARIA

## Input / Constrains

* On the **first line** of input you will get the **key** that will be used for decryption
* On the next lines until you receive “**final**” you will get lines in format:

**{encrypted teamA} {encrypted teamB} {teamA score}:{teamB score}**

* **Team scores** will be **integer numbers** in the **range [0...231]**

## Output

**League standings:**

**{place}. {TEAM NAME} {points}**

**...**

* **Print the standings table** **ordered descending by points** in format:

Where **place** is a number in range **[1… number of teams].**

**Top 3 scored goals:**

**- {team name} -> {goals}**

**- {team name} -> {goals}**

**- {team name} -> {goals}**

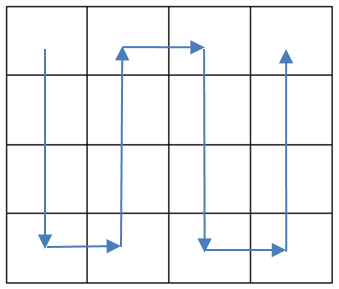
* Then you should print the top 3 team **ordered by goals in descending order** in format:
* **All team’s names** should be **uppercase.**
* **For more clarification, see the examples on the next page.**

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ??  ??ecnarF?? ??kramneD?? 0:0  ..??airagluB??32 ??dnalgnE??gf 3:2  Fg??NIAPS?? fgdrt%#$??YNAMREG??gtr 3:4  ??eCnArF?? >>??yLATi??<< 2:2  final | League standings:  1. BULGARIA 3  2. GERMANY 3  3. FRANCE 2  4. DENMARK 1  5. ITALY 1  6. ENGLAND 0  7. SPAIN 0  Top 3 scored goals:  - GERMANY -> 4  - BULGARIA -> 3  - SPAIN -> 3 |
| **Input** | **Output** |
| KZL  fdKZLairagluBKZL KZLkramneDKZLll 2:0  kzljjjKZLAiRaGluBKZL KZLylATIKZLkk 1:1  KZLkRamnedKZL KZLYlatiKZL 4:4  final | League standings:  1. BULGARIA 4  2. ITALY 2  3. DENMARK 1  Top 3 scored goals:  - ITALY -> 5  - DENMARK -> 4  - BULGARIA -> 3 |

# Problem 4. Trifon’s Quest

**Our hero Trifon** was sent on a quest. You are given **his health** and the **map** (as a **rectangle matrix**) where he will travel. Trifon **moves one cell at a time** and the quests **advances with 1 turn (first operate with health points then increase the turns)**. Print the outcome of the quest if it is **successful or not**. A quest is successful if Trifon **goes thru all cells on the map and has health above 0**.

There are different **obstacles** on his way.

* **Fight** (F) – the hero **loses {current number of turns} / 2 of his health points**
* **Heal** (H) – the hero **restores {current number of turns} / 3 of his health points**
* **Trap** (T) – the hero must **wait 2 turns** (the quest will finish 2 turns later)
* **Empty cell** (E) – nothing happens here just game **advances with 1 turn**.

**\* NOTE:** The **division of the turns** must be **integer division**. **For example:** 5 / 2 = 2.

The hero **always starts** his journey in the **top left corner** (the cell with coordinates **[0,0]**). And moves as it’s shown on **the picture**.

## Input / Constrains

* On the **first line** you will receive the **staring** **health points**. **Integer numbers** in range **[0… 231]**
* On the **second line** – **dimensions of the map** in format: “{Rows} {Cols}**”**
* On the next {Rows} **lines** -> the **map** for the journey. **Map** will **contain** **only** the symbols: F, H, T, E

## Output

* In case the player **successfully** finishes the game print:

**Quest completed!**

**Health: {Hero’s current health}**

**Turns: {Total number of turns for the quest}**

* In case the player **did not succeed** in completing the quest print:

**Died at: [{row}, {col}]**

Where **row** and **col** are the coordinates of the cell where the **health** of the hero **becomes ≤ 0**.

## Examples

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 10  4 4  FFFF  HHHH  EEEE  EEEE | Quest completed!  Health: 5  Turns: 16 |  | 8  4 3  HHF  FHH  EFT  HHT | Quest completed!  Health: 11  Turns: 16 |  | 50  4 5  TTFFF  EHHFF  THTFF  EHTTT | Died at: [1, 4] |