# JS Fundamentals Mid Exam Preparation - 1

# Problem 1 - Guinea Pig

Link: <https://judge.softuni.org/Contests/Practice/Index/2031#0>

*Merry has a guinea pig named Puppy, that she loves very much. Every month she goes to the nearest pet store and buys him everything he needs – food, hay, and cover.*

On the **first three lines**, you will receive **the quantity of food**, **hay**, and **cover**, which Merry buys for a **month (30 days)**. On the **fourth line**, you will receive the **guinea pig's weight**.

**Every day** Puppy eats **300 gr of food**. **Every** **second** day Merry **first feeds the pet**, then gives it a **certain amount of hay** **equal to** **5%** of the rest of the **food**. On **every** **third** day, Merry puts Puppy **cover** with **a quantity of** **1/3** of its **weight**.

**Calculate** whether the quantity of **food, hay, and cover**, will be enough for a **month**.

**If Merry runs out of food, hay, or cover, stop the program!**

## Input

* **On the first line – quantity food in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the second line – quantity hay in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the third line – quantity cover in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**
* **On the fourth line – guinea's weight in kilograms** - afloating-point number in the range **[0.0 – 10000.0]**

## Output

* If the food, the hay, and the cover are enough, print:
  + **"Everything is fine! Puppy is happy! Food: {excessFood}, Hay: {excessHay}, Cover: {excessCover}."**
* If one of the things is not enough, print:
  + **"Merry must go to the pet store!"**

**The output values must be formatted to the second decimal place!**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  5  5.2  1 | Everything is fine! Puppy is happy! Food: 1.00, Hay: 1.10, Cover: 1.87. |
| You receive food – **10000**, hay – **5000**, cover – **5200**, weight – **1000** (in grams).  On the first day, Merry gives Puppy 300gr food – 9700gr food left.  On the second day, the food left is **9400gr**, so the needed hay is **9400 \* 5% = 470**,and thehay left is **4530.**  On the third day, the cover left is **4866.67,** and the food left is **9100**,and so on.  On the last day, Merry has: food – 1.00, hay – 1.10, and cover – 1.87. | |
| 1  1.5  3  1.5 | Merry must go to the pet store! |
| 9  5  5.2  1 | Merry must go to the pet store! |

**JS Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["10",  "5",  "5.2",  "1"]) | Everything is fine! Puppy is happy! Food: 1.00, Hay: 1.10, Cover: 1.87 |
| (["1",  "1.5",  "3",  "1.5"  ]) | Merry must go to the pet store! |
| (["9",  "5",  "5.2",  "1"]) | Merry must go to the pet store! |

# Problem 2. Mu Online

Link: <https://judge.softuni.org/Contests/Practice/Index/2028#1>

You have **initial health 100 and initial bitcoins 0**. You will be given **a string representing the dungeon's rooms**. Each room is separated with **'|'** (vertical bar): **"room1|room2|room3…"**

Each room contains **a command** and a **number**, separated by space. The command can be:

* **"potion"**
  + You are healed with the number in the second part. But your health **cannot exceed** your **initial health (100)**.
  + First print: **"You healed for {amount} hp."**
  + After that, print your current health: **"Current health: {health} hp."**
* **"chest"**
  + You've found some bitcoins, the number in the second part.
  + Print: **"You found {amount} bitcoins."**
* In **any other case,** you are **facing a monster**, which you will **fight**. The **second part of the room** contains the **attack** of the monster. You should remove the monster's attack from your health.
  + If you are not dead (health <= 0), you've slain the monster, and you should print: **"You slayed {monster}."**
  + If you've died, print **"You died! Killed by {monster}."** and your quest is over. Print the best room you've manage to reach: **"Best room: {room}"**

If you managed to **go through all the rooms** in the dungeon, print on the **following three lines**:

**"You've made it!"**

**"Bitcoins: {bitcoins}"**

**"Health: {health}"**

### Input / Constraints

You receive a **string** representing the dungeon's rooms, separated with **'|'** (vertical bar): **"room1|room2|room3…"**.

### Output

Print the corresponding messages described above.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| rat 10|bat 20|potion 10|rat 10|chest 100|boss 70|chest 1000 | You slayed rat.  You slayed bat.  You healed for 10 hp.  Current health: 80 hp.  You slayed rat.  You found 100 bitcoins.  You died! Killed by boss.  Best room: 6 |
| **Input** | **Output** |
| cat 10|potion 30|orc 10|chest 10|snake 25|chest 110 | You slayed cat.  You healed for 10 hp.  Current health: 100 hp.  You slayed orc.  You found 10 bitcoins.  You slayed snake.  You found 110 bitcoins.  You've made it!  Bitcoins: 120  Health: 65 |

### JS Input / Output

|  |  |
| --- | --- |
| **Input** | **Output** |
| "rat 10|bat 20|potion 10|rat 10|chest 100|boss 70|chest 1000" | You slayed rat.  You slayed bat.  You healed for 10 hp.  Current health: 80 hp.  You slayed rat.  You found 100 bitcoins.  You died! Killed by boss.  Best room: 6 |
| **Input** | **Output** |
| "cat 10|potion 30|orc 10|chest 10|snake 25|chest 110" | You slayed cat.  You healed ѝfor 10 hp.  Current health: 100 hp.  You slayed orc.  You found 10 bitcoins.  You slayed snake.  You found 110 bitcoins.  You've made it!  Bitcoins: 120  Health: 65 |

# Problem 3 - Man-O-War

Link: <https://judge.softuni.org/Contests/Practice/Index/1773#2>

*The pirates encounter a huge Man-O-War at sea.*

Create a program that **tracks** the **battle** and either chooses a **winner** or prints a **stalemate**. On the **first line,** you will receive the **status** of the **pirate ship**, which is a **string** representing **integer sections** separated by **">"**. On **the second line,** you will receive the **same** type of status, but for the **warship**:

**"{section1}>{section2}>{section3}… {sectionn}"**

On the **third line,** you will receive the **maximum health capacity** a section of the ship can reach.

The following lines represent commands **until** **"Retire"**:

* **"Fire {index} {damage}"** - the pirate ship **attacks** the warship with the **given damage** at that section. Check if the **index is valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the warship **sinks**, print the following and **stop** the program: **"You won! The enemy ship has sunken."**
* **"Defend {startIndex} {endIndex} {damage}"** - the warship **attacks** the pirate ship with the **given damage** at that **range** (**indexes are inclusive)**. Check if both **indexes are valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the pirate ship **sinks**, print the following and **stop** the program:

**"You lost! The pirate ship has sunken."**

* **"Repair {index} {health}"** - the crew **repairs** a section of the **pirate ship** with the **given health**. Check if the **index is valid** and if not, **skip** the command. The health of the section **cannot** exceed the **maximum health capacity**.
* **"Status"** - prints the **count** of all sections of the **pirate ship** that need repair soon, which are all sections that are **lower than 20%** of the **maximum** **health capacity**. Print the following:

**"{count} sections need repair."**

In the end, if a **stalemate** occurs, print the **status** of **both** ships, which is the **sum** of their individual sections, in the following format:

**"Pirate ship status: {pirateShipSum}**

**Warship status: {warshipSum}"**

### Input

* On the **1st line,** you are going to receive the **status** of the **pirate ship** (**integers** separated by **'>'**)
* On the **2nd line,** you are going to receive the **status** of the **warship**
* On the **3rd line,** you will receive the **maximum health** a section of a ship can reach.
* On the following **lines**, until **"Retire"**, you will be receiving commands.

### Output

* Print the output in the **format** **described** **above**.

### Constraints

* The **section numbers** will be integers in the range [**1**….**1000**]
* The **indexes** will be integers [**-200**….**200**]
* The **damage** will be an integer in the range [**1**….**1000**]
* The **health** will be an integer in the range [**1**….**1000**]

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| 12>13>11>20>66  12>22>33>44>55>32>18  70  Fire 2 11  Fire 8 100  Defend 3 6 11  Defend 0 3 5  Repair 1 33  Status  Retire | 2 sections need repair.  Pirate ship status: 135  Warship status: 205 | |
| **Comments** | | |
| First, we receive the command "**Fire 2 11**", and damage the warship at section index 2, which is currently 33, and after reduction, the status of the warship is the following:  **12 22 22 44 55 32 18**  The **second** and **third** commands have **invalid indexes**, so we skip them.  The **fourth** command, **"Defend 0 3 5"** damages **4 sections** of the pirate ship with **5,** which results in the following states:  **7 8 6 15 66**  The **fifth** command, **"Repair 1 33"** repairs the pirate ship section and adds **33 health** to the current **8,** which results in **41**  Only **2 sections** of the pirate ship (**7** and **6**) need repair soon.  In the end, there is a **stalemate,** so we print both ship statuses (**sum** of all sections). | | |
| **Input** | | **Output** |
| 2>3>4>5>2  6>7>8>9>10>11  20  Status  Fire 2 3  Defend 0 4 11  Repair 3 18  Retire | 3 sections need repair.  You lost! The pirate ship has sunken. | |

### JS Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["12>13>11>20>66",  "12>22>33>44>55>32>18",  "70",  "Fire 2 11",  "Fire 8 100",  "Defend 3 6 11",  "Defend 0 3 5",  "Repair 1 33",  "Status",  "Retire"]) | 2 sections need repair.  Pirate ship status: 135  Warship status: 205 |
| **Comments** | |
| First, we receive the command "**Fire 2 11**", and damage the warship at section index 2, which is currently 33, and after reduction, the status of the warship is the following:  **12 22 22 44 55 32 18**  The **second** and **third** commands have **invalid indexes**, so we skip them.  The **fourth** command, **"Defend 0 3 5"** damages **4 sections** of the pirate ship with **5,** which results in the following states:  **7 8 6 15 66**  The **fifth** command, **"Repair 1 33"** repairs the pirate ship section and adds **33 health** to the current **8,** which results in **41**  Only **2 sections** of the pirate ship (**7** and **6**) need repair soon.  In the end, there is a **stalemate,** so we print both ship statuses (**sum** of all sections). | |
| **Input** | **Output** |
| (["2>3>4>5>2",  "6>7>8>9>10>11",  "20",  "Status",  "Fire 2 3",  "Defend 0 4 11",  "Repair 3 18",  "Retire"]) | 3 sections need repair.  You lost! The pirate ship has sunken. |