# JS Fundamentals Mid Exam Preparation

## 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 |
| **Comment** | |
| 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! |

## Problem 2. MuOnline

**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 managed 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 |
| "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. Memory Game

**Link:** <https://judge.softuni.org/Contests/Practice/Index/2517#2>

Write a program that recreates the **Memory game**.

On the first line, you will **receive a sequence of elements**. Each element in the sequence **will have a** **twin**. Until the player receives **"end"** from the console, you will receive **strings with two integers** separated by a space, representing **the indexes** of elements in the sequence.

If the player **tries to cheat** and enters **two equal indexes** or indexes which are **out of bounds of the sequence**, you should **add** two matching elements at the middle of the sequence in the following format:

**"-{number of moves until now}a"**

Then print this message on the console:

**"Invalid input! Adding additional elements to the board"**

**Input**

* On the **first** line**,** you will receive a **sequence of elements.**
* On the **following** lines, you will receive **integers** until the command **"end".**

**Output**

* Every time the player hit **two matching elements**, you should **remove** them from the sequence and **print** on the console the following message:

**"Congrats! You have found matching elements - {element}!"**

* If the player hit **two different elements**, you should **print** on the console the following message:

**"Try again!"**

* If the player hit **all matching elements** before he receives **"end"** from the console, you should **print** on the console the following message:

**"You have won in {number of moves until now} turns!"**

* If the player receives **"end"** **before** **he hits all matching elements**, you should **print** on the console the following message:

**"Sorry you lose :(**

**{the current sequence's state}"**

### Constraints

* All elements in the sequence will always have a matching element.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [  "1 1 2 2 3 3 4 4 5 5",  "1 0",  "-1 0",  "1 0",  "1 0",  "1 0",  "end"  ] | Congrats! You have found matching elements - 1!  Invalid input! Adding additional elements to the board  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 3!  Congrats! You have found matching elements - -1a!  Sorry you lose :(  4 4 5 5 |
| **Comment** | |
| 1)  1 0  1 1 2 2 3 3 4 4 5 5 –> 1 = 1, equal elements, so remove them. Moves: 1  2)  -1 0  -1 is invalid index so we add additional elements  2 2 3 3 -2а -2а 4 4 5 5, Moves: 2  3)  1 0  2 2 3 3 -2а -2а 4 4 5 5 -> 2 = 2, equal elements, so remove them. Moves: 3  4)  1 0  3 3 -2а -2а 4 4 5 5 -> 3 = 3, equal elements, so remove them. Moves: 4  5)  1 0  -2а -2а 4 4 5 5 -> -2а = -2а, equal elements, so remove them. Moves: 5  6)  You receive the end command.  There are still elements in the sequence, so the player loses the game.  Final state - 4 4 5 5 | |
| [  "a 2 4 a 2 4",  "0 3",  "0 2",  "0 1",  "0 1",  "end"  ] | Congrats! You have found matching elements - a!  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 4!  You have won in 3 turns! |
| [  "a 2 4 a 2 4",  "4 0",  "0 2",  "0 1",  "0 1",  "end"  ] | Try again!  Try again!  Try again!  Try again!  Sorry you lose :(  a 2 4 a 2 4 |