# JS Fundamentals Mid Exam Preparation

## Problem 1. Computer Store

**Link:** <https://judge.softuni.org/Contests/Practice/Index/2517#0>

Write a program that **prints you a receipt** for your new computer. You will receive the **parts' prices (without tax)** until you receive what type of customer this is - **special** or **regular**. Once you receive the type of customer you should print the receipt.

The **taxes are 20%** of each part's price you receive.

If the customer is **special**, he has a 10% discount on the total price with taxes.

If a given price is not a positive number, you should print **"Invalid price!"** on the console and continue with the next price.

If the total price is equal to zero, you should print **"Invalid order!"** on the console.

### Input

* You will receive numbers representing **prices (without tax)** until command **"special"** or **"regular":**

### Output

* The receipt should be in the following format:

**"Congratulations you've just bought a new computer!**

**Price without taxes: {total price without taxes}$**

**Taxes: {total amount of taxes}$**

**-----------**

**Total price: {total price with taxes}$"**

**Note: All prices should be displayed to the second digit after the decimal point! The discount is applied only on the total price. Discount is only applicable to the final price!**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ([  '1050',  '200',  '450',  '2',  '18.50',  '16.86',  'special'  ]) | Congratulations you've just bought a new computer!  Price without taxes: 1737.36$  Taxes: 347.47$  -----------  Total price: 1876.35$ |
| **Comment** | |
| 1050 – valid price, total 1050  200 – valid price, total 1250  …  16.86 – valid price, total 1737.36  We receive **special**  Price is positive number, so it is valid order  Price without taxes is 1737.36  Taxes: 20% from 1737.36 = 347.47  Final price = 1737.36 + 347.47 = 2084.83  Additional 10% discount for special customers  2084.83 – 10% = 1876.35 | |
| **Input** | **Output** |
| ([  '1023',  '15',  '-20',  '-5.50',  '450',  '20',  '17.66',  '19.30', 'regular'  ]) | Invalid price!  Invalid price!  Congratulations you've just bought a new computer!  Price without taxes: 1544.96$  Taxes: 308.99$  -----------  Total price: 1853.95$ |
| ([  'regular'  ]) | Invalid order! |

## Problem 2. Treasure Hunt

**Link:** <https://judge.softuni.org/Contests/Practice/Index/1773#1>

*The pirates need to carry a treasure chest safely back to the ship, looting along the way.*

Create a program that **manages** the **state** of the **treasure chest** along the way. On the **first line,** you will receive the **initial loot** of the treasure chest, which is a **string** of **items** separated by a **"|"**.

**"{loot1}|{loot2}|{loot3} … {lootn}"**

The following lines represent commands **until** **"Yohoho!"** which ends the treasure hunt:

* **"Loot {item1} {item2}…{itemn}"**:
  + Pick up treasure loot along the way. Insert the items at the **beginning** of the chest.
  + If an item is **already** contained, **don't** insert it.
* **"Drop {index}"**:
  + **Remove** the loot at the given **position** and **add** it at the **end** of the treasure chest.
  + If the index is **invalid,** skip the command.
* **"Steal {count}"**:
  + Someone steals the **last count** loot items. If there are **fewer items** than the given count, **remove as much** as there are.
  + Print the stolen items separated by **", "**:

**"{item1}, {item2}, {item3} … {itemn}"**

In the end, output the **average treasure gain,** which is the **sum** of all treasure items **length** divided by the **count** of all items inside the chest **formatted** to the **second decimal** point:

**"Average treasure gain: {averageGain} pirate credits."**

If the chest is **empty,** print the following message:

**"Failed treasure hunt."**

## Input

* On the **1st line,** you are going to receive the **initial treasure chest** (**loot** separated by **"|"**)
* On the following **lines**, until **"Yohoho!"**, you will be receiving commands.

## Output

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

## Constraints

* The **loot items** will be strings containing any ASCII code.
* The **indexes** will be integers in the range [**-200**…**200**]
* The **count** will be an integer in the range [**1**….**100**]

## JS Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| (["Gold|Silver|Bronze|Medallion|Cup",  "Loot Wood Gold Coins",  "Loot Silver Pistol",  "Drop 3",  "Steal 3",  "Yohoho!"]) | Medallion, Cup, Gold  Average treasure gain: 5.40 pirate credits. |
| **Comments** | |
| The first command **"Loot Wood Gold Coins"** adds **Wood** and **Coins** to the chest but **omits** Gold since it is already contained. The chest now has the following items:  **Coins Wood Gold Silver Bronze Medallion Cup**  The **second** command adds **only Pistol** to the chest  The **third** command **"Drop 3"** removes the **Gold** from the chest, but immediately adds it at the **end**:  **Pistol Coins Wood Silver Bronze Medallion Cup Gold**  The **fourth** command **"Steal 3"** removes the **last 3** items **Medallion**, **Cup**, **Gold** from the chest and prints them.  In the end calculate the average treasure gain which is the sum of all items length Pistol(**6**) + Coins(**5**) + Wood(**4**) + Silver(**6**) + Bronze(**6**) = **27** and **divide** it by the count 27 / 5 = **5.4** and format it to the **second decimal** point. | |
| **Input Output** | |
| (["Diamonds|Silver|Shotgun|Gold",  "Loot Silver Medals Coal",  "Drop -1",  "Drop 1",  "Steal 6",  "Yohoho!"]) | Coal, Diamonds, Silver, Shotgun, Gold, Medals  Failed treasure hunt. |

## Problem 3. Numbers

Link: <https://judge.softuni.org/Contests/Practice/Index/2474#2>

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

## Input

* Read from the console a single line holding **space-separated integers**.

## Output

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

## Constraints

* All input **numbers** are integers in the **range** [-1 000 000 … 1 000 000].
* The **count of numbers** is in the **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}. |