# Final Exam Preparation – 25 March 2022

## Secret Chat

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2307#0**](https://judge.softuni.org/Contests/Practice/Index/2307#0)

*You have plenty of free time, so you decide to write a program that conceals and reveals your received messages. Go ahead and type it in!*

On the first line of the input, you will receive the **concealed message**. After that, until the "Reveal" command is given, **you will receive strings with instructions for different operations that need to be performed upon the concealed message** to **interpret** **it** and reveal its actual content. There are several types of instructions, split by ":|:"

* "InsertSpace:|:{index}":
  + Inserts a single **space** **at the given index**. The given index will always be valid.
* "Reverse:|:{substring}":
  + If the message contains the given **substring**, **cut it out**, **reverse** it and **add** it at the **end** of the message.
  + If not, print "error".
  + This operation should replace only the first occurrence of the given **substring** **if there are two or more occurrences**.
* "ChangeAll:|:{substring}:|:{replacement}":
  + Changes **all occurrences** of the given **substring** with the **replacement text**.

### Input / Constraints

* On the first line, you will receive a string with a message.
* On the following lines, you will be receiving commands, split by **":|:"**.

### Output

* After each set of instructions, print the resulting string.
* After the "Reveal" command is received, print this message:  
  "**You have a new text message: {message}**"

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| heVVodar!gniV  ChangeAll:|:V:|:l  Reverse:|:!gnil  InsertSpace:|:5  Reveal | hellodar!gnil  hellodarling!  hello darling!  You have a new text message: hello darling! |
| **Comments** | |
| **ChangeAll:|:V:|:l** heVVodar!gniV -> hellodar!gnil (We replace all occurrences of "V" with "l")  **Reverse:|:!gnil**  hellodar!gnil -> !gnil -> ling! -> hellodarling! (We reverse !gnil to ling! And put it at the end of the string)  **InsertSpace:|:5**  hellodarling! -> hello.darling! (We insert a space at index 5)  Finally, after receiving the **"Reveal"** command, we print the resulting message. | |
| **Input** | **Output** |
| Hiware?uiy  ChangeAll:|:i:|:o  Reverse:|:?uoy  Reverse:|:jd  InsertSpace:|:3  InsertSpace:|:7  Reveal | Howare?uoy  Howareyou?  error  How areyou?  How are you?  You have a new text message: How are you? |

## Ad Astra

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2525#1**](https://judge.softuni.org/Contests/Practice/Index/2525#1)

*You are an astronaut who just embarked on a mission across the solar system. Since you will be in space for a long time, you have packed a lot of food with you. Create a program, which helps you identify how much food you have left and gives you information about its expiration date.*

On the first line of the input, you will be given a **text string**. You must extract the information about the food **and calculate the total calories.**

First, you must **extract the food info**. It will always follow the same pattern rules:

* It will be surrounded by "|" or "#" (only one of the two) in the following pattern:   
  #{item name}#{expiration date}#{calories}# or   
  |{item name}|{expiration date}|{calories}|
* The item name will contain **only lowercase and uppercase letters and whitespace**
* The expiration date will always follow the pattern: **"**{day}/{month}/{year}"**, where the day, month, and year will be exactly two digits long**
* The calories will be **an integer between 0-10000**

Calculate **the total calories of all food items** and then determine **how many days you can last with the food you have**. Keep in mind that **you need 2000kcal a day**.

### Input / Constraints

* You will receive a single string

### Output

* First, print **the number of days** you will be able to last with the food you have:

**"You have food to last you for: {days} days!"**

* **The output for each food item should look like this:  
  "Item: {item name}, Best before: {expiration date}, Nutrition: {calories}"**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | | |
| #Bread#19/03/21#4000#|Invalid|03/03.20||Apples|08/10/20|200||Carrots|06/08/20|500||Not right|6.8.20|5| | | |
| **Output** | | **Comments** |
| You have food to last you for: 2 days!  Item: Bread, Best before: 19/03/21, Nutrition: 4000  Item: Apples, Best before: 08/10/20, Nutrition: 200  Item: Carrots, Best before: 06/08/20, Nutrition: 500 | | We have a total of three matches – bread, apples, and carrots.  The sum of their calories is 4700. Since you need 2000kcal a day, we divide 4700/2000, which means this food will last you for 2 days.  We print each item |
| **Input** | | |
| $$#@@%^&#Fish#24/12/20#8500#|#Incorrect#19.03.20#450|$5\*(@!#Ice Cream#03/10/21#9000#^#@aswe|Milk|05/09/20|2000| | | |
| **Output** | | **Comments** |
| You have food to last you for: 9 days!  Item: Fish, Best before: 24/12/20, Nutrition: 8500  Item: Ice Cream, Best before: 03/10/21, Nutrition: 9000  Item: Milk, Best before: 05/09/20, Nutrition: 2000 | | We have three matches. The total calories are 8500 + 9000 + 2000 = 19500, which means you have food for a total of 9 days. |
| **Input** | | |
| Hello|#Invalid food#19/03/20#450|$5\*(@ | | |
| **Output** | **Comments** | |
| You have food to last you for: 0 days! | We have no matches, which means we have no food. The colored text is not a match since it doesn't have a # at the end. | |

## P!irates

**Link:** [**https://judge.softuni.org/Contests/Practice/Index/2302#2**](https://judge.softuni.org/Contests/Practice/Index/2302#2)

*Anno 1681. The Caribbean. The golden age of piracy. You are a well-known pirate captain by the name of Jack Daniels. Together with your comrades Jim (Beam) and Johnny (Walker), you have been roaming the seas, looking for gold and treasure… and the occasional killing, of course. Go ahead, target some wealthy settlements and show them the pirate's way!*

Until the "Sail" command is given, you will be receiving:

* You and your crew have targeted **cities**, with their **population** and **gold**, separated by "||".
* If you receive a city that has already been received, you have to increase the population and gold with the given values.

After the "Sail" command, you will start receiving lines of text representing events until the "End" command is given.

Events will be in the following format:

* "Plunder=>{town}=>{people}=>{gold}"
  + You have successfully attacked and plundered the town, killing the given number of people and stealing the respective amount of gold.
  + For every town you attack print this message: "{town} plundered! {gold} gold stolen, {people} citizens killed."
  + If any of those two values (population or gold) **reaches zero**, the town is disbanded.
    - You need to **remove it** from your collection of targeted cities and print the following message: **"{town} has been wiped off the map!"**
  + There will be no case of receiving more people or gold than there is in the city.
* "Prosper=>{town}=>{gold}"
  + There has been dramatic economic growth in the given city**, increasing its treasury** by the given amount of gold.
  + The gold amount **can be a negative number, so be careful.** If a negative amount of gold is given, print: "Gold added cannot be a negative number!" and ignore the command.
  + If the given gold is a valid amount, increase the town's gold reserves by the respective amount and print the following message:

"{gold added} gold added to the city treasury. {town} now has {total gold} gold."

### Input

* On the first lines, until the **"Sail"** command, you will be receiving strings representing the cities with their gold and population, separated by **"||"**.
* On the following lines, until the **"End"** command, you will be receiving strings representing the actions described above, separated by **"=>"**.

### Output

* After receiving the "End" command, if there are any existing settlements on your list of targets, you need to print all of them in the following format:

"Ahoy, Captain! There are {count} wealthy settlements to go to:

{town1} -> Population: {people} citizens, Gold: {gold} kg

{town2} -> Population: {people} citizens, Gold: {gold} kg

…

{town…n} -> Population: {people} citizens, Gold: {gold} kg"

* If there are no settlements left to plunder, print:

"Ahoy, Captain! All targets have been plundered and destroyed!"

### Constraints

* The initial population and gold of the settlements will be valid 32-bit integers, never negative, or exceed the respective limits.
* The town names in the events will always be valid towns that should be on your list.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Tortuga||345000||1250  Santo Domingo||240000||630  Havana||410000||1100  Havana||410000||1100  Sail  Plunder=>Tortuga=>75000=>380  Prosper=>Santo Domingo=>180  End | Tortuga plundered! 380 gold stolen, 75000 citizens killed.  180 gold added to the city treasury. Santo Domingo now has 810 gold.  Ahoy, Captain! There are 3 wealthy settlements to go to:  Tortuga -> Population: 270000 citizens, Gold: 870 kg  Santo Domingo -> Population: 240000 citizens, Gold: 810 kg  Havana -> Population: 410000 citizens, Gold: 1100 kg |
| **Input** | **Output** |
| Nassau||95000||1000  San Juan||930000||1250  Campeche||270000||690  Port Royal||320000||1000  Port Royal||100000||2000  Sail  Prosper=>Port Royal=>-200  Plunder=>Nassau=>94000=>750  Plunder=>Nassau=>1000=>150  Plunder=>Campeche=>150000=>690  End | Gold added cannot be a negative number!  Nassau plundered! 750 gold stolen, 94000 citizens killed.  Nassau plundered! 150 gold stolen, 1000 citizens killed.  Nassau has been wiped off the map!  Campeche plundered! 690 gold stolen, 150000 citizens killed.  Campeche has been wiped off the map!  Ahoy, Captain! There are 2 wealthy settlements to go to:  San Juan -> Population: 930000 citizens, Gold: 1250 kg  Port Royal -> Population: 420000 citizens, Gold: 3000 kg |