# Programming Fundamentals with Python: Exam Preparation

## 01. World Tour

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2518#0**](https://judge.softuni.org/Contests/Practice/Index/2518#0)**.**

*You are a world traveler, and your next goal is to make a world tour. To do that, you have to plan out everything first. To start with, you would like to plan out all of your stops where you will have a break.*

On the **first line,** you will be given a string containing all of your **stops**. Until you receive the command **"Travel"**, you will be given some commands to **manipulate** that initial string. The **commands can be**:

* **"Add Stop:{index}:{string}"**:
  + **Insert** the given **string** at that **index** only if the index **is valid**
* **"Remove Stop:{start\_index}:{end\_index}"**:
  + **Remove** the elements of the string from the **starting index** to the **end index** (**inclusive**) if **both** indices are **valid**
* **"Switch:{old\_string}:{new\_string}"**:
  + If the **old string** is in the initial string, **replace** it with the **new one** (all **occurrences**)

***Note: After each command, print the current state of the string***

After the **"Travel"** command, print the following: **"Ready for world tour! Planned stops: {string}"**

### Input / Constraints

* **JavaScript**: you will receive a **list of strings**
* An **index is valid** if it is **between the first and the last element index (inclusive)** in the sequence**.**

### Output

* Print the proper output messages in the proper cases as described in the problem description

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hawai::Cyprys-Greece  Add Stop:7:Rome  Remove Stop:11:16  Switch:Hawai:Bulgaria  Travel | Hawai::RomeCyprys-Greece  Hawai::Rome-Greece  Bulgaria::Rome-Greece  Ready for world tour! Planned stops: Bulgaria::Rome-Greece |

## 02. Ad Astra

**Submit your solutions in the SoftUni judge system at** [**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. | |

## 03. Need for Speed III

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2307#2**](https://judge.softuni.org/Contests/Practice/Index/2307#2)**.**

*You have just bought the latest and greatest computer game – Need for Seed III. Pick your favorite cars and drive them all you want! We know that you can't wait to start playing.*

On the first line of the standard input, you will receive an integer **n** – the **number of cars** that you can obtain. On the next **n** lines, the **cars themselves** will follow with their **mileage** and **fuel** **available**, separated by "|" in the following format:

"{car}|{mileage}|{fuel}"

Then, you will be receiving different **commands**, each on a new line, separated by " : ", until the "Stop" command is given:

* "Drive : {car} : {distance} : {fuel}**"**:
  + You need to **drive the given distance**, and you will **need the given** fuel to do that. If the car **doesn't have enough fuel**, print: "**Not enough fuel to make that ride**"
  + If the car has the required fuel available in the tank, **increase its mileage** with **the given distance**, **decrease its fuel with the given fuel,** and **print**:   
    "{car} driven for {distance} kilometers. {fuel} liters of fuel consumed."
  + You like driving new cars only, so if a car's mileage reaches **100 000** km, remove it from the collection(s) and print: "**Time to sell the {car}!**"
* "Refuel : {car} : {fuel}**"**:
  + **Refill** the tank of your car.
  + Each tank can hold a **maximum of 75 liters of fuel**, so if the given amount of fuel is more than you can fit in the tank, take only what is required to fill it up.
  + Print a message in the following format: "{car} refueled with {fuel} liters"
* "Revert : {car} : {kilometers}**"**:
  + Decrease the **mileage** of the given **car with the given kilometers** and print the kilometers you have decreased it with in the following format:  
    "{car} mileage decreased by {amount reverted} kilometers"
  + If the mileage becomes **less** **than** **10 000km** **after** it is decreased, **just set it to 10 000km** and   
    **DO NOT print anything.**

Upon receiving the "Stop" command, you need to print all cars in your possession in the following format:  
"**{car} -> Mileage: {mileage} kms, Fuel in the tank: {fuel} lt.**"

### Input/Constraints

* The **mileage** and **fuel** of the cars will be valid, 32-bit integers, and will never be negative.
* The **fuel** and **distance** amounts **in the commands will never be negative**.
* The **car** **names** in the **commands** will always be **valid cars in your possession**.

### Output

* All the output messages with the appropriate formats are described in the problem description.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  Audi A6|38000|62  Mercedes CLS|11000|35  Volkswagen Passat CC|45678|5  Drive : Audi A6 : 543 : 47  Drive : Mercedes CLS : 94 : 11  Drive : Volkswagen Passat CC : 69 : 8  Refuel : Audi A6 : 50  Revert : Mercedes CLS : 500  Revert : Audi A6 : 30000  Stop | Audi A6 driven for 543 kilometers. 47 liters of fuel consumed.  Mercedes CLS driven for 94 kilometers. 11 liters of fuel consumed.  Not enough fuel to make that ride  Audi A6 refueled with 50 liters  Mercedes CLS mileage decreased by 500 kilometers  Audi A6 -> Mileage: 10000 kms, Fuel in the tank: 65 lt.  Mercedes CLS -> Mileage: 10594 kms, Fuel in the tank: 24 lt.  Volkswagen Passat CC -> Mileage: 45678 kms, Fuel in the tank: 5 lt. |
| **Comments** | |
| After we receive the cars with their mileage and fuel, we start driving them. When we get to "**Drive : Volkswagen Passat CC : 69 : 8**" command, our program calculates that there is not enough fuel, and we print the appropriate message. Then we refuel the Audi A6 with 50 l of fuel and Revert the Mercedes with 500 kilometers.  When we receive the "Revert : Audi A6 : 30000", we set its mileage to **10000** km, because if the current mileage of the Audi is **38543** kms and if we subtract **30000** from it, we receive **8543** kms, which is less than 10000 kms.  After all the commands, we print our current collection of cars with their current mileage and current fuel. | |
| **Input** | **Output** |
| 4  Lamborghini Veneno|11111|74  Bugatti Veyron|12345|67  Koenigsegg CCXR|67890|12  Aston Martin Valkryie|99900|50  Drive : Koenigsegg CCXR : 382 : 82  Drive : Aston Martin Valkryie : 99 : 23  Drive : Aston Martin Valkryie : 2 : 1  Refuel : Lamborghini Veneno : 40  Revert : Bugatti Veyron : 2000  Stop | Not enough fuel to make that ride  Aston Martin Valkryie driven for 99 kilometers. 23 liters of fuel consumed.  Aston Martin Valkryie driven for 2 kilometers. 1 liters of fuel consumed.  Time to sell the Aston Martin Valkryie!  Lamborghini Veneno refueled with 1 liters  Bugatti Veyron mileage decreased by 2000 kilometers  Lamborghini Veneno -> Mileage: 11111 kms, Fuel in the tank: 75 lt.  Bugatti Veyron -> Mileage: 10345 kms, Fuel in the tank: 67 lt.  Koenigsegg CCXR -> Mileage: 67890 kms, Fuel in the tank: 12 lt. |