# Programming Fundamentals Final Exam Retake 10.04.2020

## Problem 1. Secret Chat

*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 be receiving strings** with **instructions** for different **operations** that need to be performed upon the **concealed message** in order to **interpret** **it** and reveal its true content. There are several types of instructions, split by ":|:"

* InsertSpace:|:{index}
  + Inserts a single **empty 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 more than one such 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 message.
* On the next 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 in 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? |

|  |  |
| --- | --- |
| **JavaScript 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 in 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. | |
| **JavaScript 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? |

## Problem 2. Mirror words

*The SoftUni Spelling Bee competition is here. But it`s not like any other Spelling Bee competition out there, it`s different and a lot more fun! You, of course, are a participant and you are eager to show the competition that you are the best, so go ahead, learn the rules and win!*

On the first line of the input you will be given a **text string**. In order to win the competition you have to find all hidden **word pairs**, read them and mark the ones that are **mirror** **images** of each other.

First of all you have to **extract the hidden word pairs**. Hidden word pairs are:

* Surrounded by "@" or "#" (only one of the two) in the following pattern #wordOne##wordTwo# or @wordOne@@wordTwo@
* At least **3 characters long each** (**without the surrounding symbols**)
* Made up of **letters** **only**

If the second word, **spelled backwards** is the **same** **as the first word** **and vice versa** (**casing matters**!), then they are a **match** and you have to store them somewhere. **Examples** of mirror words:

#Part##traP# @leveL@@Level@ #sAw##wAs#

* If you don`t find any valid pairs print: **"No word pairs found!"**
* If you find valid pairs print their count: **"{valid pairs count} word pairs found!"**
* If there are no mirror words print: **"No mirror words!"**
* If there are mirror words print:

"The mirror words are:

{wordOne} <=> {wordtwo}, {wordOne} <=> {wordtwo}, {wordOne} <=> {wordtwo}, etc…"

### Input / Constraints

* You will recive a string.

### Output

* Print the proper output messages in the proper cases as described in the problem description.
* If there are pairs of mirror words, print them in the end, each pair separated by **", "**.
* Each pair of mirror word must be printed with **" <=> "** between the words.

### Examples

|  |  |
| --- | --- |
| **Input** | |
| @mix#tix3dj#poOl##loOp#wl@@bong&song%4very$long@thong#Part##traP##@@leveL@@Level@##car#rac##tu@pack@@ckap@#rr#sAw##wAs#r#@w1r | |
| **Output** | **Comments** |
| 5 word pairs found!  The mirror words are:  Part <=> traP, leveL <=> Level, sAw <=> wAs | There are 5 green and yellow pairs that meet all requirements and thus are valid.  #poOl##loOp# is valid and looks very much like a mirror words pair but it isn`t because the casings don`t match.  #car#rac# “rac” spelled backwards is "car" but this is not a valid pair because there is only one "#" between the words.  @pack@@ckap@ is also valid but "ckap" backwards is "pakc" which is not the same as "pack", so they are not mirror words. |
| **Input** | |
| #po0l##l0op# @bAc##cAB@ @LM@ML@ #xxxXxx##xxxXxx# @aba@@ababa@ | |
| **Output** | **Comments** |
| 2 word pairs found!  No mirror words! | "xxxXxx" backwards is not the same as "xxxXxx"  @aba@@ababa@ is a valid pair but the word lengths are different, thus these are definitely not mirror words |
| **Input** | |
| #lol#lol# @#God@@doG@# #abC@@Cba# @Xyu@#uyX# | |
| **Output** | **Comments** |
| No word pairs found!  No mirror words! |  |

|  |  |
| --- | --- |
| **JavaScript Input** | |
| [  '@mix#tix3dj#poOl##loOp#wl@@bong&song%4very$long@thong#Part##traP##@@leveL@@Level@##car#rac##tu@pack@@ckap@#rr#sAw##wAs#r#@w1r'  ] | |
| **Output** | **Comments** |
| 5 word pairs found!  The mirror words are:  Part <=> traP, leveL <=> Level, sAw <=> wAs | There are 5 green and yellow pairs that meet all requirements and thus are valid.  #poOl##loOp# is valid and looks very much like a mirror words pair but it isn`t because the casings don`t match.  #car#rac# “rac” spelled backwards is "car" but this is not a valid pair because there is only one "#" between the words.  @pack@@ckap@ is also valid but "ckap" backwards is "pakc" which is not the same as "pack", so they are not mirror words. |
| **JavaScript Input** | |
| [ '#po0l##l0op# @bAc##cAB@ @LM@ML@ #xxxXxx##xxxXxx# @aba@@ababa@' ] | |
| **Output** | **Comments** |
| 2 word pairs found!  No mirror words! | "xxxXxx" backwards is not the same as "xxxXxx"  @aba@@ababa@ is a valid pair but the word lengths are different, thus these are definitely not mirror words |
| **JavaScript Input** | |
| [ '#lol#lol# @#God@@doG@# #abC@@Cba# @Xyu@#uyX#' ] | |
| **Output** | **Comments** |
| No word pairs found!  No mirror words! |  |

## Problem 3. Need for Speed III

*You have just bought the latest and greatest computer game – Need for Seed III. We know that you can`t wait to start playing. Pick your favorite cars and drive them all you want!*

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 the mileage of a car reaches **100 000** km, remove it from the collection(s). 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, sorted by their **mileage in decscending order**, then by their **name in ascending order**, 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  Volkswagen Passat CC -> Mileage: 45678 kms, Fuel in the tank: 5 lt.  Mercedes CLS -> Mileage: 10594 kms, Fuel in the tank: 24 lt.  Audi A6 -> Mileage: 10000 kms, Fuel in the tank: 65 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  Koenigsegg CCXR -> Mileage: 67890 kms, Fuel in the tank: 12 lt.  Lamborghini Veneno -> Mileage: 11111 kms, Fuel in the tank: 75 lt.  Bugatti Veyron -> Mileage: 10345 kms, Fuel in the tank: 67 lt. |

|  |  |
| --- | --- |
| **JavaScript 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  Volkswagen Passat CC -> Mileage: 45678 kms, Fuel in the tank: 5 lt.  Mercedes CLS -> Mileage: 10594 kms, Fuel in the tank: 24 lt.  Audi A6 -> Mileage: 10000 kms, Fuel in the tank: 65 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. | |
| **JavaScript 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  Koenigsegg CCXR -> Mileage: 67890 kms, Fuel in the tank: 12 lt.  Lamborghini Veneno -> Mileage: 11111 kms, Fuel in the tank: 75 lt.  Bugatti Veyron -> Mileage: 10345 kms, Fuel in the tank: 67 lt. |