## Problem 1 – World Tour

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/4380/programming-fundamentals-with-csharp-january-2024)

Submit your solutions in the SoftUni Judge system [here](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

* **An index is valid if it is between the first and the last element index (inclusive) (0 ….. Nth) 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 |
| Albania:Bulgaria:Cyprus:Deuchland  Add Stop:3:Nigeria  Remove Stop:4:8  Switch:Albania: Azərbaycan  Travel | AlbNigeriaania:Bulgaria:Cyprus:Deuchland  AlbNaania:Bulgaria:Cyprus:Deuchland  AlbNaania:Bulgaria:Cyprus:Deuchland  Ready for world tour! Planned stops: AlbNaania:Bulgaria:Cyprus:Deuchland |

## Problem 2 – Mirror Words

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/4380/programming-fundamentals-with-csharp-january-2024)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/2307#1)

*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**. 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 backward,** is the **same** **as the first word** **and vice versa** (**casing matters**!), 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}"**

### Input / Constraints

* **You will receive 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 backward is "car", but this is not a valid pair because there is only one "#" between the words.  @pack@@ckap@ is also valid, but "ckap" backward 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" backward is not the same as "xxxXxx"  @aba@@ababa@ is a valid pair, but the word lengths are different - these are definitely not mirror words |
| **Input** | |
| #lol#lol# @#God@@doG@# #abC@@Cba# @Xyu@#uyX# | |
| **Output** | **Comments** |
| No word pairs found!  No mirror words! |  |

## Problem 3 – The Pianist

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/4380/programming-fundamentals-with-csharp-january-2024)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/2525#2)

*You are a pianist, and you like to keep a list of your favorite piano pieces. Create a program to help you organize it and add, change, remove pieces from it!*

On the first line of the standard input, you will receive an integer **n** – the **number of pieces** you will initially have. On the next **n** lines, the **pieces themselves** will follow with their **composer** and **key**, separated by **"|"** in the following format: **"{piece}|{composer}|{key}"**.

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

* **"Add|{piece}|{composer}|{key}"**:
  + You need to **add the given piece** with the information about it to the other pieces and print:

"**{piece} by {composer} in {key} added to the collection!"**

* + If the piece **is already in the collection**, print:

"**{piece} is already in the collection!**"

* **"Remove|{piece}"**:
  + If the piece is in the collection, **remove it** and print:

**"Successfully removed {piece}!"**

* + Otherwise, print:

**"Invalid operation! {piece} does not exist in the collection."**

* **"ChangeKey|{piece}|{new key}"**:
  + If the piece is in the collection, **change its key with the given one** and print:

**"Changed the key of {piece} to {new key}!"**

* + Otherwise, print:

**"Invalid operation! {piece} does not exist in the collection."**

Upon receiving the **"Stop"** command, you need to print all pieces in your collection in the following format:  
"**{Piece} -> Composer: {composer}, Key: {key}**"

### Input/Constraints

* You will receive **a single integer** at first – **the initial number of pieces in the collection**.
* For each piece, you will receive a single line of text with information about it.
* Then you will receive multiple commands in the way described above until the command **"Stop"**.

### Output

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

### Examples

| **Input** | **Output** |
| --- | --- |
| 3  Fur Elise|Beethoven|A Minor  Moonlight Sonata|Beethoven|C# Minor  Clair de Lune|Debussy|C# Minor  Add|Sonata No.2|Chopin|B Minor  Add|Hungarian Rhapsody No.2|Liszt|C# Minor  Add|Fur Elise|Beethoven|C# Minor  Remove|Clair de Lune  ChangeKey|Moonlight Sonata|C# Major  Stop | Sonata No.2 by Chopin in B Minor added to the collection!  Hungarian Rhapsody No.2 by Liszt in C# Minor added to the collection!  Fur Elise is already in the collection!  Successfully removed Clair de Lune!  Changed the key of Moonlight Sonata to C# Major!  Fur Elise -> Composer: Beethoven, Key: A Minor  Moonlight Sonata -> Composer: Beethoven, Key: C# Major  Sonata No.2 -> Composer: Chopin, Key: B Minor  Hungarian Rhapsody No.2 -> Composer: Liszt, Key: C# Minor |
| **Comments** | |
| After we receive the initial pieces with their info, we start receiving commands. The first two commands are to add a piece to the collection, and since the pieces are not already added, we manage to add them. The third add command, however, **attempts to add a piece, which is already in the collection**, so we print a special message and don't add the piece. After that, we receive the remove command, and since the piece is in the collection, we remove it successfully. Finally, the last command says to change the key of a piece. Since the key is present in the collection, we modify its key. We receive the Stop command, print the information about the pieces, and the program ends. | |
| **Input** | **Output** |
| 4  Eine kleine Nachtmusik|Mozart|G Major  La Campanella|Liszt|G# Minor  The Marriage of Figaro|Mozart|G Major  Hungarian Dance No.5|Brahms|G Minor  Add|Spring|Vivaldi|E Major  Remove|The Marriage of Figaro  Remove|Turkish March  ChangeKey|Spring|C Major  Add|Nocturne|Chopin|C# Minor  Stop | Spring by Vivaldi in E Major added to the collection!  Successfully removed The Marriage of Figaro!  Invalid operation! Turkish March does not exist in the collection.  Changed the key of Spring to C Major!  Nocturne by Chopin in C# Minor added to the collection!  Eine kleine Nachtmusik -> Composer: Mozart, Key: G Major  La Campanella -> Composer: Liszt, Key: G# Minor  Hungarian Dance No.5 -> Composer: Brahms, Key: G Minor  Spring -> Composer: Vivaldi, Key: C Major  Nocturne -> Composer: Chopin, Key: C# Minor |