**Santa’s Gifts**

You will be given an **array of integers**, which represent the **house numbers** you should visit. The **commands** will lead you to them. If they lead you to **non-existing** places, **don’t move**.

* **Forward {numberOfSteps}**
* **Back {numberOfSteps}**
  + When you receive the “**Forward**” or “**Back**” command, you **move the given number of times** in this **direction** and **remove** the house in **this position** from your list. Also, when you receive the next command, you **continue from this position**.
* **Gift {index} {houseNumber}**
  + Enter a **new house number**, which the dwarves have left out on purpose, **at** **the** **given position** and move to its position.
* **Swap {indexOfFirst} {indexOfSecond}**
  + Santa wants to rearrange his path and **swap the order of two houses**. You will receive the **numbers of the houses**, that need to be switched and he doesn’t need to move to fulfill this command.

## Input

* On the first line you will receive the number of commands – **integer in the range [1-50]**
* On the second line you will receive the array of integers, that represent the houses, split by a single space – valid **integers in the range [1 – 500]**
* On the next n lines, you will receive the commands in the **following format**:
  + **Forward {steps}**
  + **Back {steps}**
  + **Gift {index} {value}**
  + **Swap {value1} {value2}**

## Output

* Print the **last position** and the **remaining houses** in the following format:

“Position {position}”

“{houseNumber}, {houseNumber}………, {houseNumber}”

## Constraints

* The house numbers will be valid integers in the range [1 - 1000]
* The number of commands will be a valid integer in the range [1 - 50]
* The commands will be given in the exact format as they are written above
* There will always be at least one valid command

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| 5  255 500 54 78 98 24 30 47 69 58  Forward 1  Swap 54 47  Gift 1 20  Back 1  Forward 3 | Position: 3  20, 47, 78, 24, 30, 54, 69, 58 | First, we receive the “Forward” command, the sleigh will start from the beginning – index 0. He has to move 1 step, so he will move to index 1 and delete the house number, which is stored there – 500. What is left of the list:  255 54 78 98 24 30 47 69 58  and Santa’s position is 1.  The next command is “Swap”. After it, the list looks like this:  255 47 78 98 24 30 54 69 58 and Santa’s position doesn’t change.  The “Gift” command has to insert at index 1 the house with number 20:  255 20 47 78 98 24 30 54 69 58 and move Santa to current index – 1.  The “Back” command has to move Santa back 1 step from his current position. He is at 1 position, so he has to move back to position 0, and remove the house number, which it stores:  20 47 78 98 24 30 54 69 58  The last “Forward” command will move him three steps forward from his current position, which is 0, so he goes to – 3 and removes the house:  20 47 78 24 30 54 69 58 |
| 6  50 40 25 63 78 54 66 77 24 87  Forward 4  Back 3  Forward 3  Gift 2 88  Swap 50 87  Forward 1 | Position: 3  87, 25, 88, 54, 77, 24, 50 |  |

**Santa’s New List**

Santa needs to start creating his new list for the next Christmas. Your job is to help him receive and keep the incoming information. He will receive information about the **names** of the children, the **type** of present they want (toy, candy, clothing) and the wanted **amount** in the following format:

**{childName}->{typeOfToy}->{amount}**

You can receive a command “**Remove->{childName}**”. In this case, you need to exclude the child from the new list with good children, but don’t change the information about the type of present he or she wanted and the wanted amount. Santa has already gotten the presents, so he might give them to another very good child.

When you receive the “**END**” command, you need to process it and print it, **ordered descending** by the total amount of presents for a child and **then by their names**. The format is given bellow.

**Input**

Until you receive "**END**" command you will be receiving information about the wanted presents in the following format:

**"{childName}->{typeOfPresent}->{amount}"**.

You can receive a command to remove a child from the list -> **"Remove->{childName}"**

**Output**

* Print the presents for each child, ordered descending by the total amount and then by their name, in the following format:

**Children:**

**{childName} -> {points}**

* After that print type of present and the total count for it in the following format:

**Presents:**

**{type} –> {count}**

**Constraints**

* The count of presents will always be a **valid integer in the range [0-100];**

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comment** |
| Marty->Toys->5  Sam->Candy->20  Leo->Candy->10  Leo->Toys->1  Katy->Clothes->4  Bobbie->Clothes->6  Tanya->Phone->1  Nasko->Tablet->3  END | Children:  Sam -> 20  Leo -> 11  Bobbie -> 6  Marty -> 5  Katy -> 4  Nasko -> 3  Tanya -> 1  Presents:  Toys -> 6  Candy -> 30  Clothes -> 10  Phone -> 1  Tablet -> 3 | First, we receive “Marty”, then “Sam”, then “Leo”. When we receive “Leo” for the second time, we have to add the amount the new type of present he wants to his total count of present for receiveing. We receive “Candy” as a type for the second time, we need to add the amount to the total count of that type. In the end, we print first the children, ordered descending by the total count of the presents they want and then by their names. |
| Teddy->Clothes->8  Johny->Toys->10  Freddie->Candy->30  Johny->Candy->20  Carrie->Phone->1  Carrie->Tablet->1  Carrie->Candy->10  Teddy->Toys->5  Remove->Teddy  END | Children:  Freddie -> 30  Johny -> 30  Carrie -> 12  Presents:  Clothes -> 8  Toys -> 15  Candy -> 60  Phone -> 1  Tablet -> 1 |  |