### Problem 1. Experience Gaining

*Write a program, that helps a player figure how many battles he will need to play in a battle video game, in order to unlock the next tank in the line.*

On **the first line** you will **receive the amount of experience** that is needed to unlock the tank. On **the second line** you will **receive the count of battles**. On **the** **next lines,** you will receive the experience the player can gain in every battle.

Calculate if he **can unlock** the tank. Keep in mind that he **gets 15%** more experience for every **third battle** and **10% less** for every **fifth battle**. You also need to **stop the program** as soon as he **collects the needed experience**.

If he managed to gather the experience, **print** **how many battles it took him** in the following format:

* "**Player successfully collected his needed experience for {battleCount} battles."**

If he was not able to do it, **print** the following message:

* **"Player was not able to collect the needed experience, {neededExperience} more needed."**

**Format** the needed experience to **the second decimal place**.

### Input

* On the **first line** you will receive the **needed experience** amount – a **real number** in the range [0.0….400000.0]
* On the **second line** you will receive the **count of battles** – an **integer number** in the range

[0….500]

* On the **next lines** you will receive the **experience earned per battle** – a **real number** in the range

[0.0….5000.0]

### Output

* If he **managed to gather** the experience**:**
  + **"Player successfully collected his needed experience for {battleCount} battles."**
* If he was **not able** to do it**:**
  + **"****Player was not able to collect the needed experience, {neededExperience} more needed."**

**NOTE: Format** the needed experience to **the second decimal place**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 500  5  50  100  200  100  30 | Player successfully collected his needed experience for 5 battles. |
| **Comments** | |
| The first line is the amount of the wanted experience. – **"500"**  The second line is the expected battles for which he has to collect the experience. – **"5"**  After that is the experience received for every battle:  **50 + 100 + (200 + (200 \* 15 %)) + 100 + (30 – (30 \* 10 %)) = 507**  So on the console is printed :  **"Player successfully collected his needed experience for 5 battles."** | |
| **Input** | **Output** |
| 500  5  50  100  200  100  20 | Player was not able to collect the needed experience, 2.00 more needed. |
| **Input** | **Output** |
| 400  5  50  100  200  100  20 | Player successfully collected his needed experience for 4 battles. |

### Problem 2. Friend List Maintenance

*Our player is having trouble with his friend list and some guys are disappearing without a reason so he asks you to create a program that will figure out what is going on and at the end will bring him a report.*

On **the first line** you will **receive** all his friends separated by **", "**.On **the next lines** until the **"Report"** command you will receive **commands**. The commands could be:

* **Blacklist {name}**
  + **Find** the name in the friend list and **change it** to **"Blacklisted"** and **print on the console**:
    - **"{name} was blacklisted."**
  + If the **name** is **not** in **the friend list** **print**:
    - **"{name} was not found."**
* **Error {index}**
  + **Check** if the **username** at the given index is **not "Blacklisted"** or **"Lost".** If it isn't, **change** the username to **"Lost"** and **print** on the console:
    - **"{name} was lost due to an error."**
* **Change {index} {newName}**
  + **Check** if the user at **index** position is in **range of the array.** If he is**,** **change** the **current** username with the **new one** and **print** on console:
    - **"****{currentName} changed his username to {newName}."**

After you **receive "Report"** print on the console the **count of blacklisted names**, the **count of lost names,** and the friend list separated by a **single space**.

### Input

* The **first input line** will contain the usernames that need to be stored.
* On the **next input** lines until **"Report"** you will **receive** commands.

### Output

* The output should be in the following format:
  + **"Blacklisted names: {blacklistedNamesCount}"**
  + **"Lost names: {lostNamesCount}"**
  + **"{name1} {name2} .. {nameN}"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Mike, John, Eddie  Blacklist Mike  Error 0  Error 1  Change 2 Mike123  Report | Mike was blacklisted.  John was lost due to an error.  Eddie changed his username to Mike123.  Blacklisted names: 1  Lost names: 1  Blacklisted Lost Mike123 |
| **Comments** | |
| On the first line are the names from the friendlist that need to be stored in an array.  After that the commands start to flow in. The first command finds Mike and blacklists him: **"Mike was blacklisted."**  After that **"Error 0"** failed because the name is already blacklisted and we do nothing.  **"Error 1":** John is replaced with **"Lost"** and the messange is sent to the console: **"John was lost due to an error."**  After that Mike changes his username to Mike123: **"Eddie changed his username to Mike123".**  And the report is asked for so the program ends with the shown output. | |
|  | |
| Mike, John, Eddie, William  Error 3  Error 3  Change 0 Mike123  Blacklist Eddie  Report | William was lost due to an error.  Mike changed his username to Mike123.  Eddie was blacklisted.  Blacklisted names: 1  Lost names: 1  Mike123 John Blacklisted Lost |

### Problem 3. Tanks Collector

Tom is a world of tanks player and he likes to collect premium tanks. You will **receive a list** of Tom's already owned premium vehicles **on a single line separated by ", ".** On the next **n** lines you will receive commands that could be:

* **Add, {tankName}**: Check if he already owns the wanted tank.
  + If he owns it, **print on console: "Tank is already bought"**
  + If not, **print on console: "Tank successfully bought"** and **add it** to the tank list.
* **Remove, {tankName}**: Check if he owns the tank.
  + If he owns it **print on console: "Tank successfully sold"** and **remove it** from the tank list.
  + If not **print on console: "Tank not found"**
* **Remove At, {index}**: Check if the **index** is in the range of the list.
  + If not **print on console: "****Index out of range"** and continue.
  + If it’s in range, **remove at the given index** and **print on console: "Tank successfully sold"**
* **Insert, {index}, {tankName}**: Check if the **index** is in range of the list and **check if he already owns the tank**.
  + If not **print on console: "Index out of range"** and continue.
  + If it's in range and he doesn't own the tank then **add the tank at the given index** and **print on** **console:**
    - **"Tank successfully bought"**
  + If the tank is in range and he owns it already than **print on console:**
    - **"Tank is already bought"**

After you go through all the commands you need to **print** the list **on a single line separated by ", ".**

### Input

* The **first input line** will contain the **list** of **already owned tanks**.
* The **second** **input line** will be the **number of commands** – an **integer number** in range [0…50].
* On the **next input lines** you will be receiving commands.

### Output

* As output you must print a single line containing the elements of the list, **joined** by **", "**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| T-34-85 Rudy, SU-100Y, STG  3  Add, King Tiger(C)  Insert, 2, IS-2M  Remove, T-34-85 Rudy | Tank successfully bought  Tank successfully bought  Tank successfully sold  SU-100Y, IS-2M, STG, King Tiger(C) |
| **Comments** | |
| The first command gives the tank list so its splitted and added into the list.  **"T-34-85 Rudy, SU-100Y, STG"**  The second commands gives the number of commands that will be received.  **"3"**  The Add command adds the tank to the list after the necessary checks.  **"Add, King Tiger(C)" – "Tank successfully bought"**  The Insert commands also adds the tank at the given spot after the necessary checks. **"Insert, 2, IS-2M" – "Tank successfully bought"**  The Remove command is the last one and after the checks the tank is sold.  **"Remove, T-34-85 Rudy" – "Tank successfully sold"**  After that we print the list on the console.  **"SU-100Y, IS-2M, STG, King Tiger(C)"** | |
| T 34, T 34 B, T92, AMX 13 57  4  Add, T 34  Remove, AMX CDC  Insert, 10, M60  Remove At, 1 | Tank is already bought  Tank not found  Index out of range  Tank successfully sold  T 34, T92, AMX 13 57 |