**Exercise: Lists**

Problems for exercise and homework for the ["C# Fundamentals" course @ SoftUni](https://softuni.bg/modules/57/tech-module-4-0)  
You can check your solutions here: [Judge](https://judge.softuni.bg/Contests/1211)

* **Train**

You will receive a **list of wagons** (integers) on the first line. Every integer represents **the number of passengers that are currently in each of the wagons**. On the next line, you will get **the max capacity of each wagon** (a single integer). Until you receive "**end**" you will be given two types of input:

* **Add** {**passengers**} – add a wagon to the end with the given number of passengers.
* {**passengers**} - find an existing wagon to **fit every passenger**, starting from the first wagon.

At the end **print the final state** of the train (each of the wagons, separated by a space).

**Example**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 32 54 21 12 4 0 23  75  Add 10  Add 0  30  10  75  end | 72 54 21 12 4 75 23 10 0 |
| 0 0 0 10 2 4  10  Add 10  10  10  10  8  6  end | 10 10 10 10 10 10 10 |

* **Change List**

Write a program, which reads a **list** of **integers** from the **console** and receives **commands**, which **manipulate** the list. Your program may receive the following commands:

* **Delete {element}** – delete all elements in the array, which are equal to the given element.
* **Insert {element} {position}** – insert an element and the given position.

You should stop the program when you receive the command **"end"**. Print the numbers in the array separated by a **single** whitespace.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 2 3 4 5 5 5 6  Delete 5  Insert 10 1  Delete 5  end | 1 10 2 3 4 6 |
| 20 12 4 319 21 31234 2 41 23 4  Insert 50 2  Insert 50 5  Delete 4  end | 20 12 50 319 50 21 31234 2 41 23 |

* **House Party**

Write a program that keeps track of guests, that are going to a house party. On the first line of input, you are going to receive the **number of commands** you are going to receive. On the next lines you are going to receive one of the following **messages**:

* **"{name} is going!"**
* **"{name} is not going!"**

If you receive the **first** message, you have to **add** the person if he/she **is not** in the list and if he/she is in the list print on the console: "**{name} is already in the list!"**. If you receive the second message, you have to **remove** the person if he/she **is** in the list and if not print: "**{name} is not in the list!**". At the end print all the guests.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Allie is going!  George is going!  John is not going!  George is not going! | John is not in the list!  Allie |
| 5  Tom is going!  Annie is going!  Tom is going!  Garry is going!  Jerry is going! | Tom is already in the list!  Tom  Annie  Garry  Jerry |

* **List Operations**

You will be given a **list** of **integer** numbers on the first line of input. You will be receiving operations you have to apply on the list until you receive the **"End"** command. The possible **commands** are:

* **Add {number} – add number at the end.**
* **Insert {number} {index} – insert number at given index.**
* **Remove {index} – remove at index.**
* **Shift left {count} – first number becomes last ‘count’ times.**
* **Shift right {count} – last number becomes first ‘count’ times.**

**Note: there is a possibility that the given index is outside of the bounds of the array. In that case print "Invalid index"**

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1 23 29 18 43 21 20  Add 5  Remove 5  Shift left 3  Shift left 1  End | 43 20 5 1 23 29 18 |
| 5 12 42 95 32 1  Insert 3 0  Remove 10  Insert 8 6  Shift right 1  Shift left 2  End | Invalid index  5 12 42 95 32 8 1 3 |

* **Bomb Numbers**

Write a program that **reads a sequence of numbers** and a **special bomb number** with a certain **power**. Your task is to **detonate every occurrence of the special bomb number** and according to its power - **his neighbors from left and right**. Detonations are performed from **left to right** and all detonated numbers **disappear**. Finally print the **sum of the remaining elements** in the sequence.

**Examples**

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1 2 2 4 2 2 2 9  4 2 | 12 | Special number is **4** with power **2**. After detontaion we are left with the sequence [1, 2, 9] with sum 12. |
| 1 4 4 2 8 9 1  9 3 | 5 | Special number is **9** with power **3.** After detontaion we are left with the sequence [1, 4] with sum 5. Since the 9 has only 1 neighbour from the right we remove just it (one number instead of 3). |
| 1 7 7 1 2 3  7 1 | 6 | Detonations are performed from left to right. We cannot detonate the second occurance of **7,** because its already destroyed by the first occurance. The numbers [1, 2, 3] survive. Their sum is 6. |
| 1 1 2 1 1 1 2 1 1 1  2 1 | 4 | The red and yellow numbers disappear in two sequential detonations. The result is the sequence [1, 1, 1, 1]. Sum = 4. |

* **Cards Game**

You will be given two hands of cards, which will be integer numbers. Assume that you have two players. You have to find out the winning deck and respectively the winner.

You start from the beginning of both hands. Compare the cards from the first deck to the cards from the second deck. The player, who has the bigger card, takes both cards and puts them at the **back** of his hand - **the second player’s card is last, and the first person’s card (the winning one) is before it (second to last)** and the player with the smaller card must **remove** the **card** from his deck. If both players’ cards have the same values - no one wins, and the two cards must be **removed** **from** the **decks**. The game is over, when one of the decks is left without any cards. You have to print the winner on the console and the sum of the left cards: **"{First/Second} player wins! Sum: {sum}".**

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 20 30 40 50  10 20 30 40 | First player wins! Sum: 240 |
| 10 20 30 40 50  50 40 30 30 10 | Second player wins! Sum: 50 |