# Spring Vacation Trip

*A group of friends decide to go to a trip for a few days during spring vacation. They have a certain budget. Your task is to calculate their expenses during the trip and find out if they are going to have enough money to finish the vacation.*

Create a program that calculates travelling expenses by entering the following information:

* **Days of the vacation**
* **Budget** - itsfor the whole group
* **The count of people**
* **Fuel per kilometer – the price for fuel** that their car consumes **per kilometer**
* **Food expenses per person**
* **Hotel room price for one night** – again, for **one person**

If the group **is bigger than 10**, they receive a **25% discount** from the **total hotel expenses**.

**Every day**, they **travel** some **distance** and you have to **calculate the expenses** for the **travelled kilometers**.

Every **third** and **fifth** day, they have some additional expenses, which are **40% of the current value of the expenses**.

Every **seventh** day, their expenses are reduced, because they **withdraw (receive)** a small amount of money – you can calculate it by **dividing the amount of the current expenses** by the **group of people**.

If the **expenses** **exceed the budget at some point**, stop calculating and print the following message:

"Not enough money to continue the trip"

If the **budget is enough**:

"You have reached the destination. You have {money}$ budget left."

**Print** the result formatted **2 digits** after the decimal separator.

## Input / Constraints

* **On the 1st line**, you are going to receive the days of the trip – **an integer** in the range **[1…100]**
* **On the 2nd line** – the budget – **a real number** in the range **[0.00 – 1000000.00]**
* **On the 3rd line** - the group of people – **an integer** in the rang **[1 - 50]**
* **On the 4th line** – the price for fuel per kilometer – **a real number** **[0.00 – 20.00]**
* **On the 5th line** – food expenses per person for a day – **a real number** **[0.00 – 50.00]**
* **On the 6th line** – the price for a room for one night per person – **a real number** **[0.00 – 1000.00]**
* On the next **n** lines – one for each of the days – the **travelled** **distance** in kilometers per day– **a real number** in the range **[0.00 - 1000]**

## Output

* "Not enough money to continue the trip. You need {money}$ more." –   
  if the budget is not enough
* "You have reached the destination. You have {money}$ budget left." – if it’s enough.

Print the result formatted **2 digits** after the decimal separator.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| **7**  **12000**  **5**  **1.5**  **10**  **20**  512  318  202  154  222  108  123 | You have reached the destination. You have 5083.48$ budget left. | We receive the days of the vacation, the budget, the group, the consumed fuel per kilometer, the food expenses and the price for a hotel room for one night. We must calclate the food expenses: **10 \* 5 \* 7 = 350**  And the price for the hotel for all of the nights:  **20 \* 5 \* 7 = 700**  The curent expenses are **1050**. For each day, we have to calculate the consumed fuel – **{travelledDistance} \* 1.5**  On every **3rd**  and **5th**  day we add the additional expenses:  **0.4 \* {currentExpenses}**  On every **7th** day, they **receive** some **money**:  **{currentExpense} / {groupOfPeople}**  After **we have added** the **fuel expenses for each day** and made the **other calculations**, the **expenses** have reached **8645.652**. When we divide them by **the group (5),** the result is **1729.1304**, so we have to **reduce the expenses** by that sum. The expenses become **6916.5216**. The budget is **enough**, so the result is:  "You have reached the destination. You have 5083.48$ budget left." |
| **10**  **20500**  **11**  **1.2**  **8**  **13**  100  150  500  400  600  130  300  350  200  300 | Not enough money to continue the trip. You need 465.79$ more. |  |

# Hello, France

*The budget was enough to get them to Frankfurt and they have some money left, but their final aim is to go to France, which means that they will need more finances. They’ve decided to make profit by buying items on discount from the Thrift Shop and selling them for a higher price. You must help them.*

Create a program that calculates the profit after buying some items and selling them on a higher price. In order to fulfil that, you are going to need certain data - you will receive a **collection of items** and a **budget** in the following format:

{type->price|type->price|type->price……|type->price}

{budget}

**The prices** for each of the types **cannot** **exceed** a certain **price**, which is given bellow:

|  |  |
| --- | --- |
| **Type** | **Maximum Price** |
| Clothes | 50.00 |
| Shoes | 35.00 |
| Accessories | 20.50 |

If a **price** for a certain **item** is **higher than** the **maximum** price, **don’t buy it**. Every time you **buy an item**, you have to **reduce the budget** with the value of **its** **price**. If you don’t have enough money for it, you **can’t buy it**. Buy **as much** items **as you can**.

You have to **increase** the price of **each of the items you have successfully bought with 40%.** Print the list with **the new prices** and **the profit** you will gain **from selling the items**. They need exactly **150$** for tickets for the train, so if their budget after selling the products is enough – print – "Hello, France!" and if not – "Time to go."

## Input / Constraints

* **On the 1st line** you are going to receive the **items with their prices** in the format described above **– real numbers in the range [0.00……1000.00]**
* **On the 2nd line**, you are going to be given the **budget** – **a real number** in the range **[0.0….1000.0]**

## Output

* Print the list with the bought item’s new prices, rounded 2 digits after the decimal separator in the following format:

"{price1} {price2} {price3} {price5}………{priceN}"

* Print the profit, **rounded 2 digits** after the decimal separator in the following format:

**"Profit: {profit**}"

* If the money for tickets are enough, print: "Hello, France!" and if not – "Time to go."

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Clothes->43.30|Shoes->25.25|Clothes->36.52|Clothes->20.90|Accessories->15.60  120 | 60.62 35.35 51.13  Profit: 42.03  Hello, France! | We start subtracting the valid prices from the budget:  120 – 43.40 = **76.7.**  76.7 – 25.25 = **51.45**  51.45 – 36.52 = **14.93**  14.93 is **less** than **20.90** and **15.60**, so we can’t buy either of the last two. We must increase **each price** with 40% and the new prices are: **60.62 35.35 51.13.** The profit is **42.03** and their new budget will be – what is left of the budget - **14.93 + {sum of all newPrices}.** It is enough, so we print: **Hello, France!** |
| Shoes->41.20|Clothes->20.30|Accessories->40|Shoes->15.60|Shoes->33.30|Clothes->48.60  90 | 28.42 21.84 46.62  Profit: 27.68  Time to go. |  |

# Last Stop

*The group has reached Paris and went to visit "La Louvre". They accidently found a map behind "The Wedding at Canna" painting. It had some instructions, so they have decided to follow them and see where they will lead them. Your job is to help them.*

Create a program that follows instructions in order to fulfil a quest. First, you will receive a collection of numbers – each **representing** a **painting number**. After that, you are going to be receiving **instructions**, until the "**END**" command is given.

* Change {paintingNumber} {changedNumber} – find the painting with the first number in the collection (**if it exists**) and **change** its **number** with the **second number –** {changedNumber}.
* Hide {paintingNumber} – find the painting with this value and **if it exists** and hide it (**remove** it).
* Switch {paintingNumber} {paintingNumber2} – find the given paintings in the collections **if they exist** and **switch** their places.
* Insert {place} {paintingNumber} – **insert** the painting (**paintingNumber**) **on the next place after** the givenone, **if it exists**.
* Reverse – you must **reverse** the **order** of the paintings.

Once you complete the instructions, print the numbers of the paintings on a single line, split by a space.

## Input / Constraints

* **On the 1st line**, you are going to receive the numbers of the paintings, split by a single space – **integer numbers** in the range **[1…1000]**
* **On the next lines**, you are going to receive **commands**, until you receive the "**END**" command

## Output

* Print the message you have received after the conversion of all numbers on a single line

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 115 115 101 114 73 111 116 75  Insert 5 114  Switch 116 73  Hide 75  Reverse  Change 73 70  Insert 10 85  END | 70 114 111 116 114 101 115 115 | The first command is "**Insert**". You have to insert **painting number 114** at the next index after the 5th:  115 115 101 114 73 111 **114** 116 75  The "**Switch**" will switch number **116** with **73** and the collection should look like this:  115 115 101 114 **116** 111 114 **73**  75  After receiving the"**Hide**" **command**, you must remove **75**. After that you receive "**Reverse**" and you have to reverse the whole collection. By receiving "**Change**" you have to exchange the value **73** with the value – **70.** The next "**Insert**"command is **invalid**, because there is **no 11th index** in the collection. |
| 77 120 115 101 101 97 78 88 112 111 108 101 111 110  Insert 5 32  Switch 97 78  Hide 88  Change 120 117  END | 77 117 115 101 101 78 32 97 112 111 108 101 111 110 |  |