**Exercise: Simple Loops**

Tasks for exercise in class and for homework to the course ["Programming Fundamentals and Unit Testing" @ SoftUni.](https://softuni.bg/trainings/4256/programming-fundamentals-and-unit-testing-september-2023)

**Test** your solution in **the Judge system**: <https://judge.softuni.org/Contests/4346>

## Numbers Up to 1000 Ending in 7

Write a program that prints the numbers in the range from **1 to 1000** that **end with the digit 7.**

|  |  |
| --- | --- |
| **Input** | **Output** |
| (none) | 7  17  27  …  997 |

### Guidelines

1. Create a **for loop** from 7 to 997 and check each number to see if it ends with 7. A number ends with 7 when the result of the **modulus division of the number by 10 is equal to 7**.



## Element That Equals to the Sum of Others

Write a program that reads **n integer numbers** entered by the user, and checks whether there is a number among them that is equal to the sum of all the others.

* If such an element exists, print **"Yes"** and on a new line **"Sum = " + its value**
* If there is no such element, print **"No"** and on a new line **"Diff = " + the difference between the largest element and the sum of the others** (absolute value).

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 7  3  4  1  1  2  **12**  1 | Yes  Sum = 12 | The first line of the input (7) indicates the count of numbers that will follow. 3 + 4 + 1 + 2 + 1 + 1 = 12 |
| 4  **6**  1  2  3 | Yes  Sum = 6 | 1 + 2 + 3 = 6 |
| 3  1  1  10 | No  Diff = 8 | |10 - (1 + 1)| = 8 |
| 3  5  5  1 | No  Diff = 1 | |5 - (5 + 1)| = 1 |
| 3  1  1  1 | No  Diff = 1 |  |

### Guidelines

1. Read the number n and iterate a loop up to it, where on each new line you read the number **num**.



1. Create variable named **sum** to hold the sum of **num** and **max** to keep track of the largest number.



1. Check if the sum is equal to max and print the corresponding output.



## The Old Library

Annie returns to her hometown after a very long period abroad. Upon returning home, she sees her grandmother's old library and remembers her favorite book. Help Annie by **writing a program**, where she **enters the title of the book** she's looking for (text). Until Ani doesn't **find her favorite book** or doesn't **check all the books** in the library, the program should **read the name of the next book (text) on a new line**. When you receive the text **"No More Books"**, then **all the books are checked**.

**• If Annie doesn't find the book she's looking for, print two lines:**

**o "The book you search is not here!"**

**o "You checked {number} books."**

**• If she finds the book, print one line:**

**o "You checked {number} books and found it."**

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Troy  Stronger  Life Style  Troy | You checked 2 books and found it. | The book Ani is looking for, in this case, is Troy. The first book in the library is Stronger, the second is Life Style, and the third book is the that Annie is looking for – Troy and the program ends. |
| The Spot  Hunger Games  Harry Potter  Torronto  Spotify  No More Books | The book you search is not here!  You checked 4 books. | The book Ani is searching for is **The Spot.** The first book in the library is Hunger Games, the second is Harry Potter, the third is Torronto and the fourth is Spotify. Since there are no more books in the library, the reading of names ends. Ani did not find the book she was looking for. |
| Bourne  True Story  Forever  More Space  The Girl  Spaceship  Strongest  Profit  Tripple  Stella  The Matrix  Bourne | You checked 10 books and found it. |  |

### Guidelines

1. Read the input from the console.



1. Start by making **two extra variables at the beginning**. These will keep track of **whether the book has been found or if all the books have been checked**. One of these variables should be a **counter** and needs to be an **integer with an initial value of zero**. This will help us keep count of how many books have been checked. The other variable should be a **boolean** and have **the initial value of false**.



1. Set up a **while** loop, where you will read a new book from the console **each time**, as long as there are books left in the library. Keep reading titles from the console, until you **reach the text "No More Books".**



1. If the book you're reading from the console **matches Annie's favorite book**, **update the value of the boolean** variable and **stop the loop**. Otherwise, **increase the counter by one**.



1. Depending on whether the book is found, **print the appropriate messages**.



## Exam Preparation

Write a program where Marin solves exam problems **until he receives** the message **"Enough"** from his teacher. For each solved problem, he receives a grade. **The program should stop reading data when the command "Enough" is given or when Marin receives a certain number of poor grades.**

**A poor grade is any grade that is less than or equal to 4.**

### Input

* **On the first line - number of poor grades - an integer in the range [1…5]**
* **After that, read multiple pairs of lines:**
* **Problem name - text (string)**
* **Grade - an integer in the range [2…6]**

### Output

* If Marin reaches the command **"Enough," print on 3 lines**:
* **"Average score: {average grade}"**
* **"Number of problems: {total number of problems}"**
* **"Last problem: {name of the last problem}"**
* **If he receives the specified number of poor grades:**
* **"You need a break, {number of poor grades} poor grades."**

**Format the average grade to have two decimal places after the decimal point.**

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 3  Money  6  Story  4  Spring Time  5  Bus  6  Enough | Average score: 5.25  Number of problems: 4  Last problem: Bus | The allowed number of poor grades is 3.  The first problem is called Money, and Marin's grade is 6.  The second problem is Story, and Marin's grade is 4.  The third problem is Spring Time, and Marin's grade is 5.  The fourth problem is Bus, and Marin's grade is 6.  The next command is Enough, and the program ends.  Average grade: 21 / 4 = 5.25  Number of solved problems: 4  Last problem: Bus |
| **Input** | **Output** | **Comments** |
| 2  Income  3  Game Info  6  Best Player  4 | You need a break, 2 poor grades. | The number of poor grades is 2.  The first problem is Income, and Marin's grade is 3.  The second problem is Game Info, and Marin's grade is 6.  The third problem is Best Player, and Marin's grade is 4.  Marin has reached the allowed number of poor grades; it's time for a break. |

### Guidelines

1. Read the input from the console:



1. At beginning, create four variables:
   1. **A counter for poor grades - with an initial value of 0**
   2. **A counter for solved exercises - with an initial value of 0**
   3. **The sum of all grades - with an initial value of 0**
   4. **The name of the last problem - with an initial value of an empty text**
   5. **Whether he has failed or not**



1. Create a **while** loop that continues as long as **the count of poor grades is less than the number you have read from the console**. In **each iteration** of the loop, read the **name of the problem and its grade**.



1. In case you receive the command **Enough**, change the value of **isFailed** to **false** and **end the loop**.
2. In **each iteration of the loop, add** Marin's grade to the **total sum of his grades** and **increase the grade counter**. If the **grade is lower than or equal to 4**, increase the **counter for poor grades**. Rewrite the name of the **last problem.**



1. After the loop, if **the count of poor grades** has reached the **maximum allowed**, print the necessary message:



## Vacation Savings

Jessie has decided to save money for a vacation and she wants your help to determine if she **will manage to save the required amount**. **Every day** she **either saves or spends a portion of her money**. If she wants to **spend more than she currently has**, she will **spend all of her money and end up with 0 leva**.

### Input

Read from the console:

* + **The money needed for the vacation - a real number in the range [1.00...25000.00]**
  + **Available money - a real number in the range [0.00...25000.00]**
  + **Then repeatedly read two lines:**
  + **Action type - a text with options "spend" and "save"**
  + **Amount to save / spend - a real number in the range [0.01… 25000.00]**

### Output

The program should end in the following cases:

* If for **5 consecutive days** Jessie **only spends money,** the console output should be:
* "You can't save the money."
* "{Total number of days passed}"
* **If Jessie saves enough money for the vacation**, console outputs:
* **"You saved the money for {total number of days passed} days."**

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 2000  1000  spend  1200  save  2000 | You saved the money for 2 days. | Money Needed for the Vacation: **2000**  Available Money: 1000  spend - subtract the next number from the available money  (1000 - 1200 = -200, which is less than 0  => available money= 0)  ~ consecutive days of spending = 1  - total days: 1  save - add the next number to the available money  (0 + 2000 = 2000)  ~ consecutive days of spending = 0  - total days: 2  Available Money (2000) >= Money Needed for the Vacation (**2000**) |
| 110  60  spend  10  spend  10  spend  10  spend  10  spend  10 | You can't save the money.  5 | Money Needed for the Vacation: **110**  Available Money: **60**  spend – subtract the next number from the available money (60 - 10 = 50)  ~ consecutive days of spending = 1  - total days: 1  spend – subtract the next number from the available money (50 - 10 = 40)  ~ consecutive days of spending = 2  - total days: 2  spend – subtract the next number from the available money (40 - 10 = 30)  ~ consecutive days of spending = 3  - total days: 3  spend – subtract the next number from the available money (30 - 10 = 20)  ~ consecutive days of spending = 4  - total days: 4  spend – subtract the next number from the available money (20 - 10 = 10)  ~ consecutive days of spending = 5  - total days: 5  5 consecutive days of spending => available money: 10  Available Money (10) < Money Needed for the Vacation (**110**) |
| 250  150  spend  50  spend  50  save  100  save  100 | You saved the money for 4 days. | Money Needed for the Vacation: **250**  Available Money: 150  spend - subtract the next number from the available money (150 - 50 = 100)  ~ consecutive days of spending = 1  - total days: 1  spend - subtract the next number from the available money (100 - 50 = 50)  ~ consecutive days of spending = 2  - total days: 2  save - add the next number to the available money (50 + 100 = 150)  ~ consecutive days of spending = 0  - total days: 3  save - add the next number to the available money (150 + 100 = 250)  ~ consecutive days of spending = 0  - total days: 4  Available Money (250) >= Money Needed for the Vacation (**250**) |

### Guidelines

1. Read the input from the console:



1. Make **two variables** at the start. Use them to keep **track of the number of days that have** passed and the number of days in a row that Jesse has spent money. Set both variables to have an **initial value of zero**:



1. Create a **while loop** that continues as long as Jesse's money is less than the money she needs for the trip and the counter for consecutive days is less than 5. With **each loop iteration**, read **two lines** from the console - the **first line is spend or save**, and the second line is the money Jesse saved or spent. Also, increase the day counter by 1:



1. Perform a check to determine whether Jesse is saving or spending money on a given day:
   1. If she's **saving**, add the saved money to her total and **reset the consecutive days counter**;
   2. If she's **spending**, subtract the amount she spent from her total and **increase the consecutive spending days counter**. Check if Jesse's **money has become less than** zero, if so she has no money left.
2. After the loop, check if Jesse has spent money for **five consecutive days** and print the corresponding message. Also, check if Jesse has managed to save the **required amount of money**. If she has, print the appropriate message:



## Steps

Gabi wants to start a healthy lifestyle and has set a goal to walk **10,000 steps every day**. However, on some days, she feels very tired from work and wants to go home before reaching her goal. Write a program that **reads from the console how many steps she takes** each time she goes out during the day. **When she reaches her goal**, the program should display **"Goal reached! Good job!"** and how many **steps she exceeded the goal by:** **"{difference} steps over the goal!"**

If she wants **to go home before reaching her goal**, she will enter **the command "Going home"** and **input the steps she took while going home**. After that, if she hasn't reached her goal, the program should display: **"{difference} more steps to reach goal."**

### Example Input and Output

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Input** | **Output** |
| 1000  1500  2000  6500 | Goal reached! Good job!  1000 steps over the goal! | 1500  300  2500  3000  Going home  200 | 2500 more steps to reach goal. |
| **Input** | **Output** | **Input** | **Output** |
| 1500  3000  250  1548  2000  Going home  2000 | Goal reached! Good job!  298 steps over the goal! | 125  250  4000  30  2678  4682 | Goal reached! Good job!  1765 steps over the goal! |

## Coins

Vending machine manufacturers aim to make their machines return **the least amount of coin change possible**. Write a program that **takes an amount - the change that needs to be returned**, and calculates how it can be done with the **minimum possible number of coins**.

**Hint**: Coins accepted by the vending machine: 2 leva, 1 lev, 50 stotinki, 20 stotinki, 10 stotinki, 5 stotinki, 2 stotinki, 1 stotinka

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1.23 | 4 | Our change is 1 leva and 23 stotinki. The machine returns it with 4 coins: a 1-lev coin, a 20-stotinki coin, a 2-stotinki coin, and a 1-stotinka coin. |
| 2 | 1 | Our change is 2 leva. The machine returns it with 1 coin of 2 leva. |
| 0.56 | 3 | Our change is 56 stotinki. The machine returns it with 3 coins: a coin of 50 stotinki, a coin of 5 stotinki, and a coin of 1 stotinka. |
| 2.73 | 5 | Our change is 2 leva and 73 stotinki. The machine returns it with 5 coins: a coin of 2 leva, a coin of 50 stotinki, a coin of 20 stotinki, a coin of 2 stotinki, and a coin of 1 stotinka. |

## Birthday Cake

You are invited to a 30th birthday party where the birthday person is having a large cake. However, they don't know **how many pieces the guests can take before the cake is finished**. Your task is to write a program that **calculates the number of pieces** guests have taken before the cake runs out. You will receive **the dimensions of the cake** (width and length - **integers** in the range [1...1000]), and then on each line, until you receive the command **"STOP"** or until **the cake is finished**, you will get the number of pieces guests take from it.

**Note: Each cake piece has a size of 1x1 cm.**

**Print** **one of the following lines** on the console:

**"{number of pieces} pieces are left."** - if you reach **STOP** and there are cake pieces left

**"No more cake left! You need {number of missing pieces} pieces more."**

### Example Input and Output

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 10  10  20  20  20  20  21 | No more cake left! You need 1 pieces more. | The cake has a length of **10** and a width of **10**  => The number of pieces = **10** \* **10** = **100**  1st take -> 100 - 20 = 80  2nd take -> 80 - 20 = 60  3rd take -> 60 - 20 = 40  4th take -> 40 - 20 = 20  5th take-> 20 - 21 = -1 < 0  => No more cake left, 1 piece is not enough |
| 10  2  2  4  6  STOP | 8 pieces are left. | The cake has a length of **10** and a width of **2**  => The number of pieces = **10** \* 2 = 20  1st take -> 20 - 2 = 18  2nd take -> 18 - 4 = 14  3rd take -> 14 - 6 = 8  4th take -> STOP command  => Pieces left: |

## Moving Out

On his eighteenth birthday, Jose decided to move out and rent an apartment. He packed his belongings **in boxes** and found a suitable apartment rental ad. He starts moving his luggage **in parts**, as he can't move everything at once. There is **limited free space** in his new apartment where he can arrange his belongings, so that the space is suitable for living. Write a **program that calculates the free volume of Jose's apartment after he moves his luggage**.

**Note: One box has exact dimensions: 1m x 1m x 1m.**

**Input**

The user enters the following data on separate lines

1. **Width of the free space - an integer in the range [1...1000]**
2. **Length of the free space - an integer in the range [1...1000]**
3. **Height of the free space - an integer in the range [1...1000]**
4. On the following lines **(until receiving the command "Done") - the number of boxes being moved into the apartment - an integer in the range [1...10000]**

**The program should stop reading data when the command "Done" is entered or when the free space runs out.**

**Output**

**Print** on the console **one of the following lines**:

* If you reach the **"Done"** command and there is still available space:

**"{remaining cubic meters} Cubic meters left."**

* If the available space runs out **before the "Done" command** is reached:

**"No more free space! You need {needed cubic meters} Cubic meters more."**

**Example Input and Output**

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| 10  10  2  20  20  20  20  122 | No more free space! You need 2 Cubic meters more. | 10 \* 10 \* 2 = 200 cubic meters available 20 + 20 + 20 + 20 + 122 = 202 cubic meters 200 - 202 = 2 cubic meters needed |
| 10  1  2  4  6  **Done** | 10 Cubic meters left. | 10 \* 1 \* 2 = 20 cubic meters available 4 + 6 = 10 cubic meters 20 - 10 = 10 cubic meters left |