# Exercise: Loops

Tasks for exercising from the course ["C - Essentials" @ SoftUni](https://softuni.bg/trainings/2465/c-essentials-august-2019).

Submit your code in the **judge system**: <https://judge.softuni.bg/Contests/1815/Exercise-Loops>

## Half Sum Element

Write a program that reads the **n-number of integers** entered by the user and checks if there is a number among them, **equal to the sum of all others**. If there is such an element, it prints **"Yes", "Sum =" + its value**; otherwise print **"No", "Diff =" + the difference between** the **largest** element and **the sum of the others (by absolute value).**

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Hints** |
| 7  3  4  1  1  2  **12**  1 | Yes  Sum = 12 | 3 + 4 + 1 + 2 + 1 + 1 = 12 |
| 4  **6**  1  2  3 | Yes  Sum = 6 | 1 + 2 + 3 = 12 |
| 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 |  |

## Old Books

Annie goes to her hometown after a very long period outside the country. On her way home she sees her grandmother's old library and remembers her favorite book. Help Annie write a program in which she enters the book she is looking for (text) and the capacity of the library (integer). **Until Annie finds her favorite book or checks everyone in the library, the program must read in a new line the name of each subsequent book (text).**

* **If the book has not been found print:**
* **"The book you search is not here!"**
* **"You checked {count} books."**
* **If the book is found:**
  + **"You checked {count} books and found it."**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Troy  8  Stronger  Life Style  Troy | You checked 2 books and found it. |
| The Spot  4  Hunger Games  Harry Potter  Torronto  Spotify | The book you search is not here!  You checked 4 books. |
| Bourne  32  True Story  Forever  More Space  The Girl  Spaceship  Strongest  Profit  Tripple  Stella  The Matrix  Bourne | You checked 10 books and found it. |

## Vacation

Jesse has decided to raise money for the trip **and wants you to help her figure out if she can raise the money**. She **saves or spends** some of her money **every day**. If she wants to **spend more than her available money, she will spend as much as she has and will be left with 0 leva.**

### Input

* **Vacation cost** – **real number [1.00.. .25000.00]**
* **Budget** – **real number [0.00... 25000.00]**

**Afterwards two lines:**

* **One of two actions – "spend" or "save".**
  + **Amount – real number [0.01… 25000.00]**

### Output

There are two possible outputs:

* If, **for 5 consecutive days**, Jesse **just spends,** write on the console**:**
* **"You can't save the money."**
* **"{days\_count}"**
* **If Jesse collects the money for the vacation, print:**
* **"You saved the money for {days\_count} days."**

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Hints** | |
| 2000  1000  spend  1200  save  2000 | You saved the money for 2 days. | Vacation cost: 2000  Budget: 1000  spend  1200 – the budget is 1000, so we set budget value to 0.  save  2000 – second day. | |
| 110  60  spend  10  spend  10  spend  10  spend  10  spend  10 | You can't save the money.  5 | 250  150  spend  50  spend  50  save  100  save  100 | You saved the money for 4 days. |

## Coins

Vending machine manufacturers wanted to make their machines return as **few coins as possible**. Write a program that accepts **the change** and then prints **the lowest possible coins count** to return the amount.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Hints** |
| 1.23 | 4 | 1 \* 1, 1 \* 20, 1 \* 2, 1 \* 1 |
| 2 | 1 | 1 \* 2 |
| 0.56 | 3 | 1 \* 50, 1 \* 5, 1 \* 1 |
| 2.73 | 5 | 1 \* 2, 1 \* 50, 1 \* 20, 1 \* 2, 1 \* 1 |

## Matrix

Print all the possible **2х2 matrices**, with the following conditions:

* Elements on the first line are **between [a;b],** the elements on the second are **between** **[c;d].**
* **The sum of the elements on the main diagonal is equal to the sum of the elements on the second**.
* **There must not be two equal elements on the same line**!

Read four integers **a, b, c, d.**

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Обяснения** | | | |
| 1  2  3  4 | 12  34  21  43 | 1 + 4 = 2 + 3 and 2 + 3 = 1 + 4. | | | |
| **Input** | **Output** | **Input** | **Output** | **Input** | **Output** |
| 2  4  4  5 | 23  45  32  54  34  45  43  54 | 7  8  3  5 | 78  34  78  45  87  43  87  54 | 5  7  5  6 | 56  56  65  65  67  56  76  65 |

## Number Pyramid

Write a program that reads an integer **n** entered by the user and prints a **pyramid of numbers as in the examples**:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 7 | 1  2 3  4 5 6  7 | 10 | 1  2 3  4 5 6  7 8 9 10 | 12 | 1  2 3  4 5 6  7 8 9 10  11 12 | 15 | 1  2 3  4 5 6  7 8 9 10  11 12 13 14 15 |

## Coding

Write a program that reads an integer from the console. The number of rows are as many as the number's digits. The first line corresponds to the units, the second row the tenths, the third row the hundreds of the number, etc. Each line must be printed with a **symbol that meets the following conditions**:

* Take the **digit** and add **33** to its ASCII value
* Print the symbol as many times as the value of the digit
* If the digit is **0** print **"**ZERO**"**

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Hints** | |
| 2049 | \*\*\*\*\*\*\*\*\*  %%%%  ZERO  ## | 2049 is four digit number so we print 4 lines.  9 + 33 = 42 so we print \*\*\*\*\*\*\*\*\*  4 + 33 = 37 so we print %%%%  0 so we print ZERO  2 + 33 = 35 so we print ## | |
| **Input** | **Output** | **Input** | **Output** |
| 9347439 | \*\*\*\*\*\*\*\*\*  $$$  %%%%  (((((((  %%%%  $$$  \*\*\*\*\*\*\*\*\* | 123456789 | \*\*\*\*\*\*\*\*\*  ))))))))  (((((((  ''''''  &&&&&  %%%%  $$$  ##  " |

## Equal Sums Even Odd Position

Write a program that reads **two six-digit integers** in the range from **100,000 to 300,000** from the console. Always the first number entered **will be less than the second**. Then print on the console in 1 line, separated by an interval, **all numbers that are between the two numbers read by the console and meet the following condition**:

* **the sum of the numbers of odd and even positions is equal. If there are no numbers matching the condition, no result is displayed**.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | **Output** | **Hints** | | | |
| 100000  100050 | 100001 100012 100023 100034 100045 | 1 + 0 + 0 = 0 + 0 + 1  1 + 0 + 1 = 0 + 0 + 2  And so on | | | |
| **Input** | **Output** | **Input** | **Output** | **Input** | **Output** |
| 123456  124000 | 123464 123475 123486 123497 123530 123541 123552 123563 123574 123585 123596 123640 123651 123662 123673 123684 123695 123750 123761 123772 123783 123794 123860 123871 123882 123893 123970 123981 123992 | 299900  300000 | 299970 299981 299992 | 100115  100120 |  |

## Sum Prime Non Prime

Write a program that reads integers in the range from **-2,147,483,648 to 2,147,483,647** until you receive a **"stop"** command. Find the sum of all entered **prime** numbers and the sum of all entered **complex-prime** numbers. Negative numbers cannot be simple, if a negative number is the input, the following message **"Number is negative."** Is displayed. In this case, the entered number is **ignored and not added** to either of the two amounts, and the program continues its execution, waiting for the next number to be entered.

Print two lines as output:

"Sum of all prime numbers is: {prime\_numbers\_sum}"

"Sum of all non prime numbers is: {nonprime\_numbers\_sum}"

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Hints** | |
| 3  9  0  7  19  4  stop | Sum of all prime numbers is: 29  Sum of all non prime numbers is: 13 | Prime numbers: 3 + 0 + 7 + 19  Complex numbers: 9 + 4 | |
| **Input** | **Output** | **Input** | **Output** |
| 30  83  33  -1  20  stop | Number is negative.  Sum of all prime numbers is: 83  Sum of all non prime numbers is: 83 | 0  -9  0  stop | Number is negative.  Sum of all prime numbers is: 0  Sum of all non prime numbers is: 0 |

## Train the trainers

The "Train the trainers" course is nearing completion and the final evaluation is approaching. Your job is to help the jury evaluate the presentations by writing a program that **calculates the average grade** from the presentation of each presentation by a student, and finally **the average success of all of them.**

From the console in the first line read the number of people in jury **n - integer in the interval [1… 20]**

**Then the name of the presentation - text is read separately**

For each presentation, on a new line reads **n - number of estimates(for each person in the jury)** - **real number in the interval** **[2.00… 6.00]**

After calculating the **average score** for a particular presentation, the console is printed

**"{presentation\_name} - {average\_score}."**

When the command is **"Finish"** print **"Student's final assessment is {average\_score\_for\_all}."**

All the values should be, formatted to the **second digit after the decimal point**.

### Examples

|  |  |  |  |
| --- | --- | --- | --- |
| **Input** | **Output** | **Hints** | |
| 2  While-Loop  6.00  5.50  For-Loop  5.84  5.66  Finish | While-Loop - 5.75.  For-Loop - 5.75.  Student's final assessment is 5.75. | 2 – different scores per presentation.  (6.00 + 5.50) / 2 = 5.75  (5.84 + 5.66) / 2 = 5.75  (6.00 + 5.50 + 5.84 + 5.66) / 4 = 5.75 | |
| **Input** | **Output** | **Input** | **Output** |
| 3  Arrays  4.53  5.23  5.00  Lists  5.83  6.00  5.42  Finish | Arrays - 4.92.  Lists - 5.75.  Student's final assessment is 5.34. | 2  Objects and Classes  5.77  4.23  Dictionaries  4.62  5.02  RegEx  2.88  3.42  Finish | Objects and Classes - 5.00.  Dictionaries" - 4.82.  RegEx - 3.15.  Student's final assessment is 4.32. |

## The Song of the Wheels

Sally Yashar's great-grandson has inherited a **four-digit password** safe deposit box. It is the secret of the singing carts. He has a car service and needs advertising, so he decided to make such a cart. The problem is that the password is hidden in the following task:

***"You will get a password if you know an integer, a control value is called,***

***rests in the range of 4 to 144 inclusive, but finding it might be painful. "***

Password has the following format: "**abcd"** and control value equal to **a\*b + c\*d,** however the following conditions are:

* **for a and b: a < b**
* **for c and d: c > d**

### The cart has four wheels, so the password will be the fourth number to be printed.

### If no such number is found, "No!" Is printed.

### Input

Single integer control number **M** – integer **[4 … 144]**;

### Output

* **Print the valid (аbcd) combinations.**
* **One of the two lines:**
  + - * **If the fourth combination is valid print: "Password: {а}{b}{c}{d}"**
      * **Otherwise** **"No!"**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 11 | 1291 1342 1381 1471 1532 1561 1651 1741 1831 1921 2351 2431  Password: 1471 |
| **Input** | **Output** |
| 139 | No! |

## Prime Pairs

Write a program that generates and prints on the console **four-digit numbers** in which the **first and second pair of digits form two-digit prime numbers** (an example of such a number is **1723**). The end value to which the pairs is determined by the other 2 digits given as input, which determine how much the end value is greater than the initial.

### Input

Read four lines:

* **Begin value of the first pair** **[10… 90]**
* **Begin value of the second pair[10… 90]**
* **The difference between the begin and the end value of the first pair** **[1… 9]**
* **The difference between the begin and the end value of the second pair [1… 9]**

### Output:

### Print on the console four-digit numbers in which the first two and the second two digits are prime two-digit numbers.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 10  20  5  5 | 1123  1323 |
| 10  30  9  6 | 1131  1331  1731  1931 |