# Exercises: Debugging and Troubleshooting Code

Problems for exercises and homework for the [“Programming Fundamentals Extended” course @ SoftUni](https://softuni.bg/courses/programming-fundamentals).

You can check your solutions here: <https://judge.softuni.bg/Contests/Compete/Index/420>.

## Debugging Exercise: Tricky Strings

The goal of this exercise is to practice **debugging techniques** in scenarios where a piece of code does not work correctly. Your task is to **pinpoint the 4 bugs** and **fix them** (without rewriting the entire code).

You can download the broken solution from [here](https://softuni.bg/downloads/svn/soft-tech/Jan-2017/Programming-Fundamentals-Extended-Jan-2017/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code-Exercises-Broken-Solutions.zip).

### Problem Description

You are given a **delimiter**. On the next line, you will receive a number **N.** On the next **N lines**, you will receive **strings** on each line. Your task is to **print** the strings, **separated** by the **delimiter**.

Note: the delimiter and strings could be ***anything***: whitespace and empty stringsare **acceptable input**!

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Program Output (Wrong)** | **Expected Output (Correct)** |
| -  5  I  Am  Five  Strings  Long | I-IAm-IAmFive-IAmFiveStrings-IAmFiveStringsLong-IAmFiveStringsLong- | I-Am-Five-Strings-Long |
| \_  5  you  cannot  trick  me | you\_youcannot\_youcannot\_youcannottrick\_youcannottrickme\_youcannottrickme\_ | you\_cannot\_\_trick\_me |
| '  7  S  o  f  t  U  n  i | S'So'Sof'Soft'SoftU'SoftUn'SoftUni'SoftUni' | S'o'f't'U'n'i |

### Hints

* Download the source code and get familiar with it
* Deal with poor code formatting - Remove unnecessary blank lines, indent the code properly
* Fix method parameters naming
* Give methods a proper name

## Debugging Exercise: Triangle Formations

The goal of this exercise is to practice **debugging techniques** in scenarios where a piece of code does not work correctly. Your task is to **pinpoint the bugs** and **fix them** (without rewriting the entire code).

You can download the broken solution from [here](https://softuni.bg/downloads/svn/soft-tech/Jan-2017/Programming-Fundamentals-Extended-Jan-2017/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code-Exercises-Broken-Solutions.zip).

### Problem Description

You are given **3 integer numbers: a**, **b and c**, which will represent the **3 sides of a triangle**. Your task is to check whether the triangle is **valid**.  
  
If it is, print "**Triangle is valid.**".  
Otherwise print "**Invalid Triangle.**" and **end the program**.

If it is valid, you have to check if it is a **right triangle** (a2 + b2 == c2).  
If it is a right triangle, print "**Triangle has a right angle between sides a and b**", depending on which side forms a **right angle**. If the sides **b** and **c** form a right angle, print "**Triangle has a right angle between sides b and c**", and so on.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Program Output (Wrong)** | **Expected Output (Correct)** |
| 3  4  5 | Invalid Triangle.  Triangle has a right angle between sides a and b | Triangle is valid.  Triangle has a right angle between sides a and b |
| 5  5  5 | Invalid Triangle.  Triangle has no right angles. | Triangle is valid.  Triangle has no right angles |
| 3  1  1 | Invalid Triangle.  Triangle has no right angles. | Invalid Triangle. |

### Hints

* Download the source code and get familiar with it
* Deal with poor code formatting - Remove unnecessary blank lines, indent the code properly
* Fix method parameters naming
* Give methods a proper name

## Debugging Exercise: Tetris

The goal of this exercise is to practice **debugging techniques** in scenarios where a piece of code does not work correctly. Your task is to **pinpoint the bugs** and **fix them** (without rewriting the entire code).

You can download the broken solution from [here](https://softuni.bg/downloads/svn/soft-tech/Jan-2017/Programming-Fundamentals-Extended-Jan-2017/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code-Exercises-Broken-Solutions.zip).

### Problem Description

You will receive an integer **N**. On the next several lines, you will receive commands in the form of **directions** – left, right, up and down

Your task is to print the famous **Tetris** block – the **T-shape**, facing the given **direction**, with a **size** of **N**.

When you receive the command “exit”, you should stop the program execution.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Program Output (Wrong)** | **Expected Output (Correct)** |
| 1  left  right  exit | *(no output)* | **.\***  **\*\***  **.\***  **\*.**  **\*\***  **\*.** |
| 2  up  down  exit | \*\*  \*\*  \*\*  \*\*  \*\*  \*\*  \*\*  *(infinite loop)* | **..\*\*..**  **..\*\*..**  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **..\*\*..**  **..\*\*..** |
| 3  right  left  exit | \*\*\*\*\*\*  \*\*\*\*\*\*  .\*.  ......  ......  \*\*\*\*\*\*  \*\*\*\*\*\* | **\*\*\*...**  **\*\*\*...**  **\*\*\*...**  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **\*\*\*...**  **\*\*\*...**  **\*\*\*...**  **...\*\*\***  **...\*\*\***  **...\*\*\***  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **\*\*\*\*\*\***  **...\*\*\***  **...\*\*\***  **...\*\*\*** |

### Hints

* Download the source code and get familiar with it
* Deal with poor code formatting - Remove unnecessary blank lines, indent the code properly
* Fix method parameters naming
* Give methods a proper name

## Debugging Exercise: Mining Coins

The goal of this exercise is to practice **debugging techniques** in scenarios where a piece of code does not work correctly. Your task is to **pinpoint the bugs** and **fix them** (without rewriting the entire code).

You can download the broken solution from [here](https://softuni.bg/downloads/svn/soft-tech/Jan-2017/Programming-Fundamentals-Extended-Jan-2017/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code/05.%20Programming-Fundamentals-Debugging-and-Troubleshooting-Code-Exercises-Broken-Solutions.zip).

### Problem Description

In this problem you have to decrypt **encrypted data** and **calculate** its **value**. You will receive an **integer** N. On the next **N** lines, you will receive 3-digit integers.

The input integers will be **indexed** - starting from **1 to N**. The integers represent **characters** of an **encrypted message**.

You must take the **first** and the **last** digit and form a **2-digit number** from them, depending on the **current index** of the input number:

* if the index is an **odd** number – **subtract** the **middle digit** from the **2-digit number**.
* If the index is **even** number – **add** the **middle digit** to the **2-digit** number.

If the resulting number is an **English** **alphabet** **letter** (Lower or Uppercase) **append** it to a string. If it is not, **ignore** it.

After that, calculate the **sum** of the **3 digits** and **divide** it by **N**.

The result from this operation should be a **floating-point number**, **rounded** to the **7-th digit** after the decimal point.

Do the same for **all input integers** and **sum** the results. The sum is the **value** of the **decrypted data**.

The final result should be the **decrypted string** (the letters, appended together), and the **value**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Program Output (Wrong)** | **Expected Output (Correct)** |
| 8  836  736  733  795  650  778  886  694 | Message:  Value: 15.25000000 | **Message: SOFTUNI**  **Value: 17.6250000** |
| 7  618  811  918  918  716  716  710 | Message: aa  Value: 12.71429000 | **Message:** **CRacKME**  **Value: 13.8571400** |

### Hints

* Download the source code and get familiar with it
* Deal with poor code formatting - Remove unnecessary blank lines, indent the code properly
* Fix method parameters naming
* Give methods a proper name