**More Exercise: Basic Syntax, Conditional Statements and Loops**

Problems for exercises and homework for the ["Programming HYPERLINK "https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023" Fundamentals" course @ HYPERLINK "https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023"SoftUni HYPERLINK "https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023".](https://softuni.bg/trainings/3951/programming-fundamentals-with-java-january-2023)

You can check your solutions in [Ju HYPERLINK "https://judge.softuni.org/Contests/1461/"d HYPERLINK "https://judge.softuni.org/Contests/1461/"ge](https://judge.softuni.org/Contests/1461/).

* **Sort Numbers**

Read three real numbers and sort them in descending order. Print each number on a new line.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  1  3 | 3  2  1 |
| -2  1  3 | 3  1  -2 |
| 0  0  2 | 2  0  0 |

* **English Name of the Last Digit**

Write a **method** that returns the **English name** of the last digit of a given number. Write a program that reads an integer and prints the returned value from this method.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 512 | two |
| 1 | one |
| 1643 | three |

* **Gaming Store**

Write a program that helps you buy the games. The **valid games** are the following games in this table:

|  |  |
| --- | --- |
| **Name** | **Price** |
| OutFall 4 | $39.99 |
| CS: OG | $15.99 |
| Zplinter Zell | $19.99 |
| Honored 2 | $59.99 |
| RoverWatch | $29.99 |
| RoverWatch Origins Edition | $39.99 |

On the first line, you will receive your **current balance** – a **floating-point** number in the range **[0.00…5000.00]**.

Until you receive the command "**Game Time**", you have to keep **buying games**. When a **game** is **bought**, the user's **balance** decreases by the **price** of the game.

Additionally, the program should obey the following conditions:

* If a game the user is trying to buy is **not present** in the table above, print "**Not Found**" and **read the next line**.
* If at any point, the user has **$0** left, print "**Out of money!**" and **end the program**.
* Alternatively, if the user is trying to buy a game that they **can't afford**, print "**Too Expensive**" and **read the next line**.

When you receive "**Game Time**", **print** the user's **remaining money** and **total spent on games**, **rounded** to the **2nd decimal place**.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 120  RoverWatch  Honored 2  Game Time | Bought RoverWatch  Bought Honored 2  Total spent: $89.98. Remaining: $30.02 |
| 19.99  Reimen origin  RoverWatch  Zplinter Zell  Game Time | Not Found  Too Expensive  Bought Zplinter Zell  Out of mo-ney! |
| 79.99  OutFall 4  RoverWatch Origins Edition  Game Time | Bought OutFall 4  Bought RoverWatch Origins Edition  Total spent: $79.98. Remaining: $0.01 |

* **Reverse String**

Write a program that reverses a string and prints it on the console.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| Hello | olleH |
| SoftUni | inUtfoS |
| 1234 | 54321 |

* **Messages**

Write a program that emulates **typing an SMS**, following this guide:

|  |  |  |
| --- | --- | --- |
| **1** | **2**  abc | **3**  def |
| **4**  ghi | **5**  jkl | **6**  mno |
| **7**  pqrs | **8**  tuv | **9**  wxyz |
|  | **0**  space |  |

Following the guide, **2** becomes "**a**", **22** becomes "**b**" and so on.

**Examples**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |  | **Input** | **Output** |
| 5  44  33  555  555  666 | hello | 9  44  33  999  0  8  44  33  777  33 | hey there | 7  6  33  33  8  0  6  33 | meet me |

**Hints**

* A native approach would be just putting all the possible combinations of digits in a giant **switch** statement.
* A cleverer approach would be to come up with a **mathematical formula** that **converts** a **number** to its **alphabet** representation:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Digit** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| * **Index** | * 0 1 2 | * 3 4 5 | * 6 7 8 | * 9 10 11 | * 12 13 14 | * 15 16 17 18 | * 19 20 21 | * 22 23 24 25 |
| * **Letter** | * a b c | * d e f | * g h i | * j k l | * m n o | * p q r s | * t u v | * w x y z |

* Let's take the number **222** (**c**), for example. Our algorithm would look like this:
* Find the **number of digits** the number has "e.g. **222** -> **3 digits**"
* Find the **main digit** of the number "e.g. **222** -> **2**"
* Find the **offset** of the number. To do that, you can use the formula: **(main digit - 2) \* 3**
* If the main digit is **8 or 9**, we need to **add 1** to the **offset** since the digits **7** and **9** have **4 letters each**
* Finally, find the **letter index** (a -> 0, c -> 2, etc.). To do that, we can use the following formula: **(offset + digit length - 1)**.
* After we've found the **letter index**, we can just add that to **the ASCII code** of the lowercase letter "**a**" (97)