# Exercise: Syntax, Functions and Statements

Problems for in-class lab for the ["JavaScript Essentials" course @ SoftUni](https://softuni.bg/courses/js-essentials). Submit your solutions in the SoftUni judge system at <https://judge.softuni.bg/Contests/1422>

1. **Road Radar**

Write a function that determines whether a driver is within the speed limit. You will receive the speed and the area. Each area has a different limit:

* On the **motorway** the limit is **130 km/h**
* On the **interstate** the limit is **90 km/h**
* In the **city** the limit is **50 km/h**
* Within a **residential** area the limit is **20 km/h**

If the driver is **within the limits**, there should not be any output. If the driver is **over the limit**, however, your function should print the severity of the infraction.

For speeding up to **20** km/hover the limit, speeding should be printed

For speeding up to **40** km/h over the limit, excessive speeding should be printed

For anything else, reckless driving should be printed

The **input** comes as an **array of elements**. The first element is the current speed (**number**), the second element is the area.

The **output** should be printed on the console. Note that in certain cases there isn’t any output.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| [40, 'city'] |  |
| [21, 'residential'] | speeding |
| [120, 'interstate'] | excessive speeding |
| [200, 'motorway'] | reckless driving |

1. **Cooking by Numbers**

Write a program that receives a **number** and a **list** of five operations. Perform the operations **sequentially** by starting with the **input number** and using the result of every operation as starting point for the next one. Print the result of every operation in order. The operations can be one of the following:

* **chop** - divide the number by two
* **dice** - square root of number
* **spice** - add 1 to number
* **bake** - multiply number by 3
* **fillet** - subtract 20% from number

The **input** comes as an **array of 6 string elements**. The first element is the starting point and must be **parsed** to a number. The remaining 5 elements are the names of the operations to be performed.

The **output** should be printed on the console.

**Examples**

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['32', 'chop', 'chop', 'chop', 'chop', 'chop'] | 16 8 4 2 1 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| ['9', 'dice', 'spice', 'chop', 'bake', 'fillet'] | 3  4  2  6  4.8 |

1. **Validity Checker**

Write a program that receives two points in the format **[x1, y1, x2, y2].** Check if the distance between each point and the start of the cartesian coordinate system (0, 0) is **valid**. A distance between two points is considered **valid**, if it is an **integer value**.

In case a distance is valid, print"{x1, y1} to {x2, y2} is valid**"**

If the distance is invalid, print **"**{x1, y1} to {x2, y2} is invalid**"**

The order of comparisons should always be first **{x1, y1}** to **{0, 0}**, then **{x2, y2}** to **{0, 0}** and finally **{x1, y1}** to **{x2, y2}**.

The **input** consists of two points given as an **array of numbers**.

For each comparison print either "{x1, y1} to {x2, y2} is valid**"** if the distance is valid, or **"**{x1, y1} to {x2, y2} is invalid**"** if it is invalid.

|  |  |
| --- | --- |
| **Input** | **Output** |
| [3, 0, 0, 4] | {3, 0} to {0, 0} is valid  {0, 4} to {0, 0} is valid  {3, 0} to {0, 4} is valid |
| [2, 1, 1, 1] | {2, 1} to {0, 0} is invalid  {1, 1} to {0, 0} is invalid  {2, 1} to {1, 1} is valid |

**Examples**