# Function Description

**Function Name:** validateWeight

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| weight | int | The weight value entered by the user to validate. |

**Returns:** int, 1 (True) if the weight is within the valid range [MIN\_WEIGHT, MAX\_WEIGHT], False otherwise.

**Description:**

* This function checks if the given weight value falls within the acceptable range.
* Algorithm: Checks if the weight falls within the acceptable range of [MIN\_WEIGHT, MAX\_WEIGHT].
* Special Conditions: The weight value must be an integer.

**Function Name:** validateBoxSize

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| boxSize | double | The size of the box entered by the user to validate. |

**Returns:** int, 1 (True) if the box size is one of the valid sizes [QUARTER\_CUBIC\_METER, HALF\_CUBIC\_METER, ONE\_CUBIC\_METER], False otherwise.

**Description:**

* This function verifies if the provided box size is valid based on predefined valid sizes.
* Algorithm: Verifies if the box size is one of the valid sizes [QUARTER\_CUBIC\_METER, HALF\_CUBIC\_METER, ONE\_CUBIC\_METER].
* Special Conditions: The box size value must be a double.

**Function Name:** validateLocation

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| map | const struct Map\* | The map structure representing the grid. |
| destination | const struct Point\* | The destination point to validate. |

**Returns:** int, 1 (True) if the destination point is within the map's boundaries, False otherwise.

**Description:**

* This function checks if the given destination point is a valid location within the provided map's boundaries.
* Algorithm: Checks if the given destination point is a valid location within the provided map's boundaries and if it is not surrounded by eight ones.
* Special Conditions: The destination point must be within the map's boundaries, and the surrounding conditions of the destination point will affect the output.

**Function Name:** validateVolume

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| truck | const struct Truck\* | The truck structure containing the boxes. |

**Returns:** int, 1 (True) if the truck's volume is within the valid range, False otherwise.

**Description:**

* Validates the volume of a truck.
* Algorithm: Checks if the truck's total volume is within the valid range (not exceeding MAX\_VOLUME).
* Special Conditions: The truck structure must contain the boxes.

**Function Name:** calculateLimitingFactor

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| truck | const struct Truck\* | The truck structure for which to calculate the limiting factor. |

**Returns:** double, the limiting factor as a percentage (weight or volume).

**Description:**

* Calculates the limiting factor for a truck (weight or volume).
* Algorithm: Compares the truck's total weight and total volume to determine the limiting factor as a percentage.
* Special Conditions: The truck structure must contain the boxes.

**Function Name:** getUserInput

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| N/A | N/A | N/A |

**Returns:** -

**Description:**

* Gets user input for shipment weight, box size, and destination.
* Algorithm: Prompts the user to enter the weight, box size, and destination, and validates the input using the defined validation functions. Continues taking input until the user enters "0 0 x".
* Special Conditions: The user must enter valid input, and the validation functions will affect the output.

**Function Name:** findAvailableTruck

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| input | struct Input\* | The input struct containing the map and trucks. |
| weight | int | The weight of the shipment in kilograms. |
| size | double | The size of the box required in cubic meters. |
| destination | struct Point | The destination of the box as a Point (row, col). |

**Returns:** A pointer to the available truck, or NULL if no truck is available.

**Description:**

* Finds an available truck for the shipment based on weight, size, and destination.
* Algorithm: Searches for a truck that meets the criteria (weight, size, and closest route to the destination). Considers the limiting factor (weight or volume) and the distance to the destination to determine the truck.
* Special Conditions: The input struct must contain the map and trucks. The weight, size, and destination must be provided, and the trucks' routes and capacities will affect the output.

**Function Name:** Name of function

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** Return type and description of what it means and special conditions that affect it.

**Description:** A description of what the function does, any special algorithms used and special condition that the user needs to be aware of that will affect the output. There needs to be sufficient detail in the description to allow the black box tests to be written before the code is complete. This description can also be given to the programmers and provide them with everything they need to know to write the code.