# Function Description

**Function Name:** calCapacity

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| truck | struct Truck | This parameter contains information about loaded boxes |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:**

* Return type: struck Size
* This function returns weight& volume capacity
* Conditions are implemented to assess whether loaded boxes have reached their maximum capacity. If no items are loaded, it will return the full capacity; if the loaded boxes have not reached their limit, it will return the calculated capacity. Finally, if the capacity is full, it will return 0 for both weight and volume capacity.

**Description:** It calculates the remaining capacity of a truck based on the loaded weight and volume of the items in the truck. This function provides a way to determine the remaining capacity of a truck based on its loaded weight and volume, considering different scenarios.

**Function Name:** validateDestination

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| route | Pointer to the constant struct Route | Passing pointer to the constant struct Route route which is shipment route |
| input | struct Shipment | This parameter contain customerinput(destination) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:**

* Return type: int
* This function returns 1 if customer input has valid destination, 0 if customer input has invalid destination.
* This function iterates through the points in the provided route and checks if any of them matches the destination specified in the input shipment. If a match is found, it sets valid to 1, indicating a valid destination; otherwise, it remains 0, indicating an invalid destination. The final return value is either 1 (valid) or 0 (invalid).

**Description:** This function essentially checks whether the destination specified in the input shipment exists in the provided route. If a match is found, it sets valid to 1, indicating a valid destination, and returns this status. If no match is found, it returns 0, indicating an invalid destination.

**Function Name:** validateBox

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| input | struct Shipment | Customer input that has information of box size |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:**

* Return type: int
* This function returns invalid code that indicate the reason of invalidation.
* This function produces the variable invalidCode, signifying the validity status of the box. It assesses both the weight and volume of the box, employing conditions to verify each. The weight must be within the range of 1 to MAX\_WEIGHT (inclusive), while the volume is expected to correspond to one of the predefined BOX\_SIZEs.
* 0 indicates both weight and volume are valid.

1 indicates if the weight is invalid.

2 indicates volume is invalid

**Description:** This function provides a simple way to check the validity of the weight and volume of a box within a shipment.

**Function Name:** findTruck

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| size | struct Size | Customer input that has information of weight |
| truck | struct Truck | Represent the shipment with its weight |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:**

* Return type: int
* This function returns result that indicate the reason of validation
* This function returns 1 if the truck is not overloaded. Otherwise return 0.

**Description:** This function provides a simple way to check the validity of the capacity of the truck

**Function Name:** bestRoute

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| Routes[MAX\_ROUTE] | Struct Route\* | Pointer array for route structures representing different paths |
| input | Struct Shipment | Constains the destination coordinates (row and column) of the shipment |
| size | int | The number of valid paths in the path array |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:**

* Return type: int
* This function calculates the distance from several paths (struct Route\* routes [MAX\_ROUTE]) to a given destination and returns the index of the path with the shortest distance.
* shortDistance: A variable representing the shortest distance, initially set to INT\_MAX. This variable stores the shortest distance among the paths found so far.
* shortIndex: A variable that represents the index of the path with the shortest distance, initially set to -1. This variable stores the index of paths with the shortest distance among the paths found so far.
* route: A variable that stores the distance from the starting point of the current path to the destination. Call the distance function to calculate this value.
* **Description:** This function represents the optimal route with the shortest distance.