# Function Description

**Function Name:** validatePackage

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| p | struct Package | A package that need to be checked for its validity |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** returns 0 or 1 mimicking boolean values. 1 representing a valid package, 0 otherwise.

**Description:** Checks whether the parameter package meets the requirements

# Function Description

**Function Name:** selectClosetRouteToDest

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| destination | struct Point | A validated destination that a truck need to make a delivery to |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** returns a struct Route type representing the closest route to the destination. It should be one of the three routes.

**Description:** Select the starting route which is closest to customer destination for a truck

# Function Description

**Function Name:** truckLessFull

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| firstTruck | struct Truck\* | A pointer to a Truck type that holds information about its current weight and capacity |
| secondTruck | struct Truck\* | A pointer to a Truck type that holds information about its current weight and capacity |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** int representing true(1) or false(0) based on whether first truck has more available weight

**Description:** The function will compare the limiting factor percentages of the two trucks to determine which one has more space remaining. The implementation of this function would involve calculating the limiting factor percentages for each truck and then comparing them to determine which truck has more space remaining. If there is no limiting factor, then the total percentage of available space is compared.

# Function Description

**Function Name:** validDestination

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| Map | struct map\* | A pointer to map information |
| Destination row | int | The row of the destination |
| Destination column | char | The column of the destination |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** int representing true(1) if destination is valid or false(0) if destination is invalid.

**Description:** This function will compare input data (row and column) with destination array. We will use for loop to check the multiple destination. If input data is same as the one of data within destination array, we will output true(1). Otherwise, we will keep output false(0), and notice user that invalid input destination and please enter again.

# Function Description

**Function Name:** compareRouteLengths

**Parameter List:**

|  |  |  |
| --- | --- | --- |
| Parameter Name | Type | Description |
| route1 | **struct Route\*** | The first route to compare. |
| route2 | **struct Route\*** | The second route to compare. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Returns:** int

**Description:** Returns -1 if route1 is shorter, 1 if route2 is shorter, or 0 if both routes are of equal length. This function compares the lengths of two routes by counting the number of points in each route. As movement can only be horizontal or vertical, each move counts as a single step.