**[Win23-SFT221-NFF-4](https://activity-1-11.atlassian.net/browse/WSN4" \t "_self)**

**F1- int isValidPackage (int weight, double size);**

Test case for minimum valid package weight and size:

weight = 1, size = 0.001, expected output = 1

Test case for maximum valid package weight and size:

weight = 100, size = 1.0, expected output = 1

Test case for invalid package weight:

weight = 0, size = 0.01, expected output = 0

Test case for invalid package size:

weight = 10, size = -0.1, expected output = 0

Test case for weight and size both zero:

weight = 0, size = 0, expected output = 0

Test case for weight as maximum integer value:

weight = INT\_MAX, size = 0.5, expected output = 0

Test case for size as maximum double value:

weight = 20, size = DBL\_MAX, expected output = 0

Test case for weight and size as non-numeric values:

weight = "abc", size = "xyz", expected output = 0

Test case for weight as negative value:

weight = -5, size = 0.1, expected output = 0

Test case for size as zero and weight greater than zero:

weight = 10, size = 0, expected output = 0

**F2- struct Point convertDestinationNametoPoint (const char destName[]);**

Test case for valid destination name:

destName = "New York", expected output = Point(40.7128, -74.0060)

Test case for invalid destination name:

destName = "Atlantis", expected output = Point(-1, -1)

Test case for empty destination name:

destName = "", expected output = Point(-1, -1)

Test case for destination name with only spaces:

destName = " ", expected output = Point(-1, -1)

Test case for destination name with leading and trailing spaces:

destName = " Paris ", expected output = Point(48.8566, 2.3522)

Test case for destination name with special characters:

destName = "Los Angeles!", expected output = Point(34.0522, -118.2437)

Test case for destination name with mixed case:

destName = "LoNdOn", expected output = Point(51.5074, -0.1278)

Test case for destination name with multiple words:

destName = "Rio de Janeiro", expected output = Point(-22.9068, -43.1729)

Test case for destination name with non-ASCII characters:

destName = "München", expected output = Point(48.1351, 11.5820)

Test case for destination name with leading/trailing numbers:

destName = "123 Sydney 456", expected output = Point(-33.8651, 151.2099)

**F3- char\* convertPointToDestinationName (const struct Point pt);**

Test case for valid point in the USA:

pt = Point(40.7128, -74.0060), expected output = "New York"

Test case for valid point in Europe:

pt = Point(48.8566, 2.3522), expected output = "Paris"

Test case for valid point in Australia:

pt = Point(-33.8651, 151.2099), expected output = "Sydney"

Test case for valid point in Asia:

pt = Point(35.6895, 139.6917), expected output = "Tokyo"

Test case for valid point in Africa:

pt = Point(-26.2041, 28.0473), expected output = "Johannesburg"

Test case for valid point in South America:

pt = Point(-23.5505, -46.6333), expected output = "São Paulo"

Test case for point with non-ASCII characters:

pt = Point(48.1351, 11.5820), expected output = "München"

Test case for point at the North Pole:

pt = Point(90.0, 0.0), expected output = "North Pole"

Test case for point at the South Pole:

pt = Point(-90.0, 0.0), expected output = "South Pole"

Test case for point with maximum double values:

pt = Point(DBL\_MAX, DBL\_MAX), expected output = "Unknown Destination"

**F4- int reachedDestination(const struct Point p1, const struct Point p2);**

Test case for the truck just reaching the destination:

p1 = Point(40.7128, -74.0060), p2 = Point(40.7129, -74.0060), expected output = 1

Test case for the truck not reaching the destination:

p1 = Point(48.8566, 2.3522), p2 = Point(40.7128, -74.0060), expected output = 0

Test case for the truck being at the same position as the destination:

p1 = Point(40.7128, -74.0060), p2 = Point(40.7128, -74.0060), expected output = 1

Test case for the truck being very far away from the destination:

p1 = Point(-33.8651, 151.2099), p2 = Point(40.7128, -74.0060), expected output = 0

Test case for the truck being very close to the destination:

p1 = Point(51.5074, -0.1278), p2 = Point(51.5074, -0.1277), expected output = 1

Test case for the truck being at the North Pole:

p1 = Point(90.0, 0.0), p2 = Point(90.0, 0.1), expected output = 1

Test case for the truck being at the South Pole:

p1 = Point(-90.0, 0.0), p2 = Point(-90.0, 0.1), expected output = 1

Test case for the truck being at the equator:

p1 = Point(0.0, 0.0), p2 = Point(0.0, 1.0), expected output = 1

Test case for the truck being in the eastern hemisphere:

p1 = Point(48.8566, 2.3522), p2 = Point(-33.8651, 151.2099), expected output = 0

Test case for the truck being in the western hemisphere:

p1 = Point(40.7128, -74.0060), p2 = Point(-33.8651, 151.2099), expected output = 0

**F5- Truck\* getTrucksWithCapacity (struct Truck trucks[], int numberOfTrucks, int weight, double size);**

Test case for when there are no trucks available:

trucks = [], numberOfTrucks = 0, weight = 10, size = 0.5, expected output = NULL

Test case for when all trucks are below capacity:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 100, size = 5.0, expected output = NULL

Test case for when some trucks have exactly enough capacity:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 200, size = 10.0, expected output = [Truck(2, 600, 25), Truck(3, 700, 30)]

Test case for when only one truck has enough capacity:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 250, size = 12.5, expected output = [Truck(3, 700, 30)]

Test case for when all trucks have enough capacity:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 400, size = 20.0, expected output = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)]

Test case for when the weight is zero:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 0, size = 5.0, expected output = NULL

Test case for when the size is zero:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 50, size = 0, expected output = NULL

Test case for when the weight is very large:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 1000, size = 50.0, expected output = NULL

Test case for when the size is very large:

trucks = [Truck(1, 500, 20), Truck(2, 600, 25), Truck(3, 700, 30)], numberOfTrucks = 3, weight = 100, size = 500.0, expected output = NULL

Test case for when numberOfTrucks is zero:

trucks = [], numberOfTrucks = 0, weight = 10, size = 0.5, expected output = NULL