**Blackbox test cases MS3**

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

**Test Cases:**

**Minimum Valid Package Weight and Size:**

**weight = 1, size = 0.001**

**Expected output: 1**

**Maximum Valid Package Weight and Size:**

**weight = 100, size = 1.0**

**Expected output: 1**

**Invalid Package Weight:**

**weight = 0, size = 0.01**

**Expected output: 0**

**Invalid Package Size:**

**weight = 10, size = -0.1**

**Expected output: 0**

**Weight and Size Both Zero:**

**weight = 0, size = 0**

**Expected output: 0**

**Weight as Maximum Integer Value:**

**weight = INT\_MAX, size = 0.5**

**Expected output: 0**

**Size as Maximum Double Value:**

**weight = 20, size = DBL\_MAX**

**Expected output: 0**

**Weight and Size as Non-Numeric Values:**

**weight = "abc", size = "xyz"**

**Expected output: 0**

**Weight as Negative Value:**

**weight = -5, size = 0.1**

**Expected output: 0**

**Size as Zero and Weight Greater Than Zero:**

**weight = 10, size = 0**

**Expected output: 0**

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

**Test Cases:**

**Valid Destination Name:**

**destName = "New York"**

**Expected output: Point(40.7128, -74.0060)**

**Invalid Destination Name:**

**destName = "Atlantis"**

**Expected output: Point(-1, -1)**

**Empty Destination Name:**

**destName = ""**

**Expected output: Point(-1, -1)**

**Destination Name with Only Spaces:**

**destName = " "**

**Expected output: Point(-1, -1)**

**Destination Name with Leading and Trailing Spaces:**

**destName = " Paris "**

**Expected output: Point(48.8566, 2.3522)**

**Destination Name with Special Characters:**

**destName = "Los Angeles!"**

**Expected output: Point(34.0522, -118.2437)**

**Destination Name with Mixed Case:**

**destName = "LoNdOn"**

**Expected output: Point(51.5074, -0.1278)**

**Destination Name with Multiple Words:**

**destName = "Rio de Janeiro"**

**Expected output: Point(-22.9068, -43.1729)**

**Destination Name with Non-ASCII Characters:**

**destName = "München"**

**Expected output: Point(48.1351, 11.5820)**

**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 Cases:**

**Valid Point in the USA:**

**pt = Point(40.7128, -74.0060)**

**Expected output: "New York"**

**Valid Point in Europe:**

**pt = Point(48.8566, 2.3522)**

**Expected output: "Paris"**

**Valid Point in Australia:**

**pt = Point(-33.8651, 151.2099)**

**Expected output: "Sydney"**

**Valid Point in Asia:**

**pt = Point(35.6895, 139.6917)**

**Expected output: "Tokyo"**

**Valid Point in Africa:**

**pt = Point(-26.2041, 28.0473)**

**Expected output: "Johannesburg"**

**Valid Point in South America:**

**pt = Point(-23.5505, -46.6333)**

**Expected output: "São Paulo"**

**Point with Non-ASCII Characters:**

**pt = Point(48.1351, 11.5820)**

**Expected output: "München"**

**Point at the North Pole:**

**pt = Point(90.0, 0.0)**

**Expected output: "North Pole"**

**Point at the South Pole:**

**pt = Point(-90.0, 0.0)**

**Expected output: "South Pole"**

**Point with Maximum Double Values:**

**pt = Point(DBL\_MAX, DBL\_MAX)**

**Expected output: "Unknown Destination"**

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

**Test Cases:**

**Truck Reaching the Destination:**

**p1 = Point(40.7128, -74.0060), p2 = Point(40.7129, -74.0060)**

**Expected output: 1**

**Truck Not Reaching the Destination:**

**p1 = Point(48.8566, 2.3522), p2 = Point(40.7128, -74.0060)**

**Expected output: 0**

**Truck Being at the Same Position as the Destination:**

**p1 = Point(40.7128, -74.0060), p2 = Point(40.7128, -74.0060)**

**Expected output: 1**

**Truck Being Very Far Away from the Destination:**

**p1 = Point(-33.8651, 151.2099), p2 = Point(40.7128, -74.0060)**

**Expected output: 0**

**Truck Being Very Close to the Destination:**

**p1 = Point(51.5074, -0.1278), p2 = Point(51.5074, -0.1277)**

**Expected output: 1**

**Truck Being at the North Pole:**

**p1 = Point(90.0, 0.0), p2 = Point(90.0, 0.1)**

**Expected output: 1**

**Truck Being at the South Pole:**

**p1 = Point(-90.0, 0.0), p2 = Point(-90.0, 0.1)**

**Expected output: 1**

**Truck Being at the Equator:**

**p1 = Point(0.0, 0.0), p2 = Point(0.0, 1.0)**

**Expected output: 1**

**Truck Being in the Eastern Hemisphere:**

**p1 = Point(48.8566, 2.3522), p2 = Point(-33.8651, 151.2099)**

**Expected output: 0**

**Truck Being in the Western Hemisphere:**

**p1 = Point(40.7128, -74.0060), p2 = Point(-33.8651, 151.2099)**

**Expected output: 0**

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

**Test Cases:**

**No Trucks Available:**

**trucks = [], numberOfTrucks = 0, weight = 10, size = 0.5**

**Expected output: NULL**

**All Trucks 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**

**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)]**

**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)]**

**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)]**

**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**

**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**

**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**

**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**

**Number of Trucks is Zero:**

**trucks = [], numberOfTrucks = 0, weight = 10, size = 0.5**

**Expected output: NULL**