Graphs and Sorting - Exercise

Problem 1: Hill-Free Route

A truck carrying a load starts from a specific location and is required to reach a destination. Due to some issues in the truck, it is not in an appropriate condition to climb up hills. Given an m × n matrix representing a route map with indication of hills (0) and plains (1), develop a function (hasHillFreeRoute) to check if there is a hill-free route from a source to a destination. Note: in the route map, the top indicates north. From any location, the truck can go in any of the 8 directions around it.

Function Prototype: int hasHillFreeRoute(int routeMap[][100], int m, int n, int src[2], int dst[2]);

Sample:

Input: routeMap =

1	0	0
0	1	0
0	0	0
1	1	0
1	0	1

m = 5, n = 3
src = {3,0}
dst = {4,2}

Output: 1 // indicates route exists {3,0} -> {3,1} -> {4,2}

Input: routeMap =

1	1	0
0	1	0
0	0	0
1	1	0
1	0	1

m = 5, n = 3
src = {3,0}
dst = {0,2}

Output: 0 // indicates route does not exist

Problem 2: Same Number of Edges to and from Node

Given the adjacency matrix of a directed graph, implement a function (sameToFro) to find the number of nodes which have the same number of edges to and from them.

Function Prototype: int sameToFro(int adjMat[][100], int n);

Sample:

Input:

1 0 1 0
1 1 0 1
0 1 0 1
0 1 0 0

Output: 2 // nodes 1 (having 2 edges) and 2 (having 3 edges)

Input:

0 0 1
1 0 1
1 0 1

Output: 0

Problem 3: Sort Points

Given an array of points P1, P2, ..., Pn with the coordinates (x1, y1), (x2, y2), ..., (xn, yn) respectively, arrange the points in an increasing order of

a. x-coordinates
b. y-coordinates
c. (x-coordinate + y-coordinate) / 2

Function Prototype: void arrangePoints(struct point A[], int start, int end);

Perform the arrangement using the following techniques:

- Merge-sort
- Quicksort

Problem 4:

An algorithm is likely to be input with positive integers too huge to be accommodated even in the highest data type of any programming language. Hence, these numbers are input as strings. Devise a non-increasing sorting algorithm to sort an array of 'n' such strings.

Function Declaration: void sortIntegerStrings(char * hugeIntegers[], int n){}

Sample:

Input: hugeIntegers = {"4", "232315982349723847238761239478497198 74", "1", "32", "132312431241341343242341324123 4123", "232314982349723847238761239478497198 74", "132312431241341343242341324123 4122"}, n = 7

Output: {"1", "4", "32", "132312431241341343242341324123 4122", "132312431241341343242341324123 4123", "232314982349723847238761239478497198 74", "232315982349723847238761239478497198 74"}

Problem 5: Modified versions of Merge Sort

Modify the merge-sort algorithm in the following ways to sort an array of 'n' elements and analyse the time complexity:

a) divide the array into 2 sub arrays, one containing only 2 elements and the other containing the remaining elements. If the size of the array is 2 or less, a constant time sorting technique is to be implemented to sort the elements.
b) divide and conquer strategy is applied only to the left sub-array, while bubble-sort is applied on the right sub-array