Problem 1

You want to go from your home to RMIT campus. Besides those two places, there are other places in the street network. The street network is model as an adjacency matrix **distance** where row i and column j (i != j) can contain:

distance[i][j] = -1, means you cannot go from place i to place j directly
distance[i][j] = X (X > 0), means you can go from place i to place j directly, and the distance is X

Note that the street network is a one-way network. If the number of places is N (including your home and RMIT campus), your home has an index 0, and RMIT campus has an index N – 1. Here is an example:

```
[
 [0, -1, 5, 10],
 [-1, 0, 4, 2],
 [-1, 1, 0, 4],
 [3, -1, 7, 0]
]
```

Explanation: you can go from your home (place 0) to place 2 (distance = 5), place 3 (distance = 10). Place 3 is also the RMIT campus. You can assume that there is always a path from your home to RMIT campus.

Create a class StreetNetwork (remember to prefix it by your first name). The constructor of StreetNetwork accepts a 2D array of integers. This array will be used in the public methods described next.

Except for the constructor, your StreetNetwork class must support the following two public operations (feel free to add additional private operations as needed). Furthermore, let's call N the number of places in the StreetNetwork instance, you must write the big-O of the public methods regarding N in the worst case (no explanation is needed).

- int nearestNeighbour(): return the index of the nearest neighbour place of your home. A place Y is a neighbour of place X if there is a positive distance from X to Y, i.e., distance[X][Y] > 0. In the example array above, this method returns 2 because the distance from your home to the neighbour place 2 is 5, which is the minimum.

- int shortestToSchool(): display the shortest path from your home to RMIT campus and return the distance of that shortest path. In the example above, the shortest path is "0 -> 2 -> 1 -> 3" and its total distance is 5 + 1 + 2 = 8.

Your code for this problem must be stored in a single file **<your first name>StreetNetwork.java**.

Problem 2

You have an array of Students sorted by GPAs. All GPAs are unique. You want to create a function upperBound(double searchGPA) to find the student whose GPA is smallest and strictly greater than the parameter searchGPA. This function returns the index of that student. If there is no student matching the condition, this function returns N (N is the number of Students).

For example, assume the GPAs of the students are = [2.0, 2.5, 2.8, 2.9, 3.1, 3.3]

Calling upperBound(1.0) return 0 (the student at index 0, whose GPA is 2.0, which is the smallest and greater than the search GPA, which is 1.0.

Calling upperBound(2.9) return 4 (the student at index 4, whose GPA is 3.1, which is the smallest and greater than the search GPA, which is 2.9.

Calling upperBound(3.3) return 6 (no student has a GPA strictly > 3.3).

Below is the pseudocode for upperBound() function

```
int upperBound(Student[] Std, double searchGPA)
      N = length(Std)
      for i = 0 to (N - 1)
            if (Std[i].GPA > searchGPA) return i
      return N
```

What is the time complexity of the above algorithm regarding N? Provide your analysis.
Propose a better algorithm for the above problem. Analyze your proposed algorithm.