Areté
DISCOVER. DEVELOP. DELIVER.

Complete the following exercises. You may choose to complete the exercises in C, C++, or Python. When coding your solutions try to make them as robust as possible and handle any edge cases or possible exceptions that could occur. You will be asked to review and/or demonstrate your solution during the interview. Note that this is not a pass/fail scenario, but rather an opportunity for us to understand how you solve problems. Specifically, we'll be looking at the following:

- Correctness: We want to see that you understood the problem and found a solution that works for all possible inputs.
- Code Quality: We are building systems that should stand the test of time, and as such need to be maintainable. If your solution is difficult to understand, it will be difficult to evaluate and maintain.

1. The provided stores.json files contains a list of stores with x-position (x), y-position (y) and name (name). Your task is to create a function that will return the N nearest stores.
    a. Your solution should take as input a path to the json file, the number of nearest stores to return (N), and an input position (x,y). The solution should return the N nearest store names.

2. The previous exercise assumes a straight path between two points. Many times, however, there are obstacles that we need to get around to get from point A to point B. In addition to the stores.json file, you have been provided the stores_map.txt file. This file represents a grid containing the store locations within its bounds along with some obstacles. (Empty spaces are marked as '.', obstacles are marked as 'X'. For your convenience, there is also a second map file that includes the redundant information of where stores are located, marked as 'S'). Your task is to create a pathfinder that finds the shortest path between an input position (x,y) and every store in the given list, and then use this to update your solution to problem 1.
    a. Your solution should take as input the path to the stores json file, the path to the map file, the number of nearest stores to return (N), and a starting position (x,y). The new solution should return the N nearest store names, using the updated path lengths as the distance between the starting position and the stores (instead of the straight-line distance, as in problem 1).