*Please note this document and the example provided is for education purposes only, for the sole use of the coding challenge exercise at the Amazon Careers Festival, not to be shared outside this environment.

Amazon Coding Challenge

OVERVIEW

In this challenge you are going to implement a pathfinding algorithm for Amazon's drone delivery service. You will need to create a path on a 2D grid that contains a starting point, a delivery point, and a number of no-fly-zones where the drone can't fly at all. The drone can move in four cardinal directions on the grid (up, down, left, right), not diagonally.

For this challenge the coordinates (0, 0) indicate the top-left edge of the grid. Test cases will be given for every task.

TASK 1

Implement a function that will take the grid size, a starting point, and a delivery point, and return a shortest valid path from the starting point to the delivery point. Include the start and delivery point in the returned path. The grid and points are zero-indexed. In some scenarios there could be multiple shortest paths, the function can return just one of them.

```
grid_size: 3
start: (0, 1); goal: (2, 2))
Returns -> [(0,1), (1,1), (2,1), (2,2)] // or other valid path
```

TASK 2

Extending the code you wrote in Task 1, implement a functionality for "no-fly-zones", which are groups of cells that the drone can't fly in. These will be provided as a list of coordinates. You can assume there will always be at least one valid path.

```
grid_size: 3
start: (0, 2); goal: (2, 2)
no_fly_zones: [(1, 1), (1, 2)]
Returns -> [(0, 2), (0, 1), (0, 0), (1, 0), (2, 0), (2, 1), (2, 2)]
```

TASK 3

There is a new requirement for drones that will fly over buildings in busy city centers. The function will be given a 2D grid of numbers, which represent the height of the buildings in the city, and a start and delivery point. Building heights are integers in the range 1-20.

There is a cost to change the altitude of the drone, equal to the difference between building heights the drone is going between. For example, flying from a building of height 4 to a building of height 6 has a cost of 2. The drone has to always fly at exactly the level of the building it's currently above, it cannot stay at a higher altitude. For example, going from a building of height 8 to a building of height 7 has a cost of 1.
Going between two buildings of the same height has no cost. The drone cannot fly over buildings over 10 units high.

Find a path through the city between the start and end point which minimizes the cost. Return the cost for the cheapest path. You can assume there will always be at least one valid path.

```
grid: [[4, 7, 6, 5], [5, 6, 5, 4], [10, 7, 12, 3], [2, 8, 7, 2]]
start: (0, 0); goal: (3, 3)
Returns -> 6
```