## Programming Assignment 3

## COP3503

# A sense of the Future

In the year 2200, a futuristic city relies on autonomous delivery robots to transport packages across different buildings. Each robot has a limited travel range before it must return to the charging station. Each building is located at a certain distance from the robot dispatch center, and a robot can only serve a building if it has enough travel capacity to reach it.

Your task is to develop a **greedy algorithm** that efficiently assigns robots to buildings, ensuring that the maximum number of deliveries are completed before the robots run out of travel distance. Each robot can deliver to **multiple buildings** as long as it still has enough energy**.** You do not need to worry about reserving extra energy for the robot to return to the charging station; it has a reserved energy supply for that purpose.

Your program will read from two text files that define the available robots and the buildings they need to deliver to.

For this assignment, you must follow these requirements.

1. You must create your solution inside a class called `GreedyRobots`.
2. You have been provided with 5 text files in a zip folder called `"InputTextFiles.zip"`.

    - Each text file contains numerical values representing distances.
    - Files named robots represent the maximum travel range of each robot.
    - Files named buildings represent the distance of each building from the dispatch center.
    - The number in the filename corresponds to a test case.
        - Example: `robots1.txt` and `buildings1.txt` represent test case 1.

3. Your class should have the following attributes:

    - An integer array that stores the maximum travel range of each robot.
    - An integer array that stores the distance of each building from the dispatch center.
    - An integer representing the number of buildings that successfully received deliveries.
    - An integer representing the number of buildings that were left unserved.

4. The `GreedyRobots` class has an overloaded constructor with four parameters:

    - The first parameter represents the number of available robots.
    - The second parameter represents the number of buildings that need deliveries.
    - The third parameter represents the name of the text file containing the maximum travel range of each robot.

- The forth parameter represents the name of the text file containing the distances of each building.

5. The `GreedyRobots` class, has a public non static method named `readFiles()`

   - It reads the respective text files (`robotsX.txt` and `buildingsX.txt`).
   - It stores the values in their respective arrays.

6. The GreedyRobots class, has a public non static method named assignDeliveries() this method will solve the problem of this assignment and it must:

   - Implements a greedy algorithm to maximize the number of deliveries.
   - The method must run in O(RM) time complexity, where R represents the number of robots and M represents the number of buildings.
   - Your solution must follow greedy techniques covered in the lecture.

7. The GreedyRobots class, has a public non static method named `displayResults()`. This method will display the results computed by assignDeliveries(). Simply, you will display the number of successful and unserved deliveries. Each result should be displayed on a separate line.

   Example Output:

```
Successful Deliveries: 8
Unserved Buildings: 2
```

## Assumption you can make about this program:

- If a robot delivers to Building 1 and still has enough energy to reach Building 2, it can proceed with the delivery to Building 2 without considering the distance between Building 1 and Building 2. The **only** constraint is that the robot must have sufficient remaining energy to reach the next building from the dispatch center

## Submission Guidelines

1. Submit your Java file (`GreedyRobots.java`) to Canvas
2. Your code must follow the coding style provided with the assignment.
3. Your code must work on Eustis
4. All text files **must be** in the same directory as your **driver file** and **solution file**. **DO NOT** create subfolders! Placing files in subdirectories will prevent our batch grader from running your code correctly, which will result in point deductions that cannot be

disputed. You can assume that all moves in the text files are valid and that there are no invalid characters.

5. Ensure that your comment header follows the correct format, similar to the Styling Guide, and is placed at the very top of your file. Failure to do so, including misplacing the header or formatting it incorrectly, will result in point deductions as outlined in the rubric.