# Multiply Labs Software Technical Challenge

## How does this challenge work?

The challenge is composed of two parts:

1. The “take-home” technical assignment (this challenge)
    - You should aim to complete this part within 4 hours and email us back your response as soon as you are done.
    - All of the information you should need for this challenge is included in this interview packet.
    - If you need to make any assumptions about this challenge, please state what assumptions you made.
    - We don’t expect you to fully complete every part of the challenge. Prioritize working on whichever parts you feel will best show off your strengths!
2. Multiply Labs internal review of your technical challenge
    - We will review your technical challenge. This review will consist of the following:
        - Reviewing your implementation of the questions.
        - Reviewing how you define your data structures.
        - Reviewing how you test your code to ensure it works as expected.
        - How intuitive it is to run your code and call specific functions.
        - We will actually run your code, which may include us calling specific functions you define.
        - **Make sure your code is runnable!**

**You must use python for this challenge (unless otherwise stated)**.

**You will submit:**

- A zip file that contains all of the code you wrote for this challenge.

Good luck with the challenge and have fun!

# Challenge:

You may use all of the contents in this zip folder as well as any information in the rest of this document to help you answer the questions below. **See the write-up below for more information on how the system works.**

## Questions:

**You are required to answer Questions 1, 2, and 3.** If you have additional time, feel free to work on Question 4 or Question 5.

### Question 1:

Given a file that defines the process by which a GRex moves around the cluster, create some data structures to ingest the file.

**Supplemental Material:**
There are multiple ways to define a process and move a GRex through the cluster. Below are two potential process files that define how a GRex moves about the cluster.

- See “grex_process_1.csv”
  - This file contains the operations, the time of those operations, and the associated modules that each GRex needs to move to within the cluster
- See “grex_process_2.csv”
  - This file contains the operations, the time of those operations, and the associated modules that each GRex needs to move to within the cluster

### Question 2:

Given a process file, analyze the following data points:

- How long it takes for a process to run to completion
- How much downtime each module has
  - Downtime is defined as the amount of time that a module is not being used when compared to the entire length of the process.

### Question 3:

Create a job-scheduling algorithm that takes in two process files and schedules the tasks of the robotic cluster. This job-scheduling algorithm should be able to parallelize operations as well as respect the precedence required by some of the operations. **The goal of this algorithm should be to minimize the amount of time spent running the cluster.**

There are a few constraints when it comes to the cluster that the algorithm must be aware of. The first is a precedence constraint, and the second are cluster constraints.

**Precedence Required:**

- The robotic arm must pick the GRex, then place that same GRex before being able to pick and place another grex. The “pick” of the GRex and the “place” of the GRex are two separate operations, but they must happen in sequence.
- All operations in the process must happen in sequence.
- Operations of different processes may happen in parallel with one another.

**Cluster constraints:**
- There is only one type of each module available in the robotic cluster.
- Only one GRex can be in each module at a time.
- If a module is not being used by one GRex, it may be used to store/hold another GRex.

**Supplemental Material:**
- See “grex_process_1.csv”
  - This file contains the operations, the time of those operations, and the associated modules that each GRex needs to move to within the cluster
- See “grex_process_2.csv”
  - This file contains the operations, the time of those operations, and the associated modules that each GRex needs to move to within the cluster

## Question 4:

Expand the functionality of the job-scheduling algorithm to take in any arbitrary number of process files and schedule them.

## Question 5:

Design a user interface (using any language or tools you would like) that allows a user to interact with this job scheduling algorithm and see relevant data. ***Please note***: We do not expect you to design a full UI. Sketches, drawings, and pseudo code are acceptable.

# Background Information:

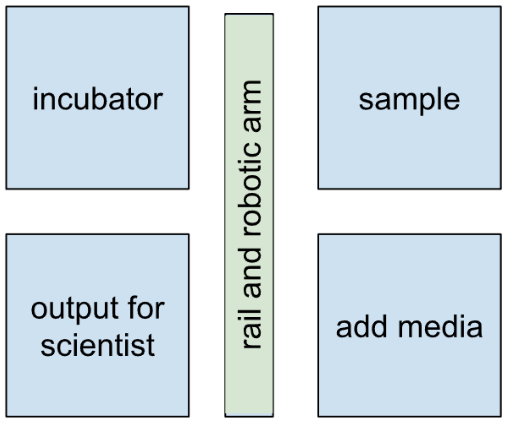
## About the System:

If you are curious, you can find more information of our system here:

- https://multiplylabs.com/products/cell-therapy/

## About the GRex Movement:

Here at Multiply Labs we grow cells using a GRex that is modified for robotic use. A GRex is a container that holds cells and media and it moves throughout the cluster. See the image below for the layout of the robotic cluster and all of the potential modules the GRex can be in.



The GRex always starts in the *"Output for Scientist"* module and it always ends in the *"Output for Scientist"* module. The rail and the robotic arm is able to pick and place the GRex from one module to another; this is how it moves around the cluster.

Each module can only house one GRex at a time and each module performs actions of a specified duration on the GRex. All module actions that are performed on the GRex have an associated operation code, which are defined in the "human_readable_information.csv" file.

As previously mentioned, the two files that are sample processes for the GRex moving around the cluster are "grex_process_1.csv" and "grex_process_2.csv". Each process represents the modules that the GRex must be in for specific operations to happen.