**PaymentWorks Coding Problem**
*for Engineering Candidates*

We include technical challenges in the recruiting process at PaymentWorks because *one aspect* of what makes an engineer successful here is technical competency. We are also strong believers that it takes more than technical expertise to make a great engineer, and we pride ourselves on making sure that challenges like this one are only part of the evaluation criteria. Other characteristics include cultural fit and a demonstrated desire and ability to take risks and continually learn from your mistakes.

That said, technical ability is important, and this challenge is useful to us in seeing more than just whether or not you can solve the problems, but additionally your style of coding and style of presenting your solution - which differs widely from developer to developer.

This problem is fairly straightforward: it is not designed to be a brainteaser or especially difficult. There is no time limit (though it shouldn't take too long), and you can code it in any development environment comfortable for you. We do this to make this process as comfortable as possible for you, while still making sure the exercise provides meaningful information.

**Input / Output Expectations:** The means by which your program accepts input and returns responses is up to you, but should at least consider the interface experience for the end-user. Examples would be accepting commands/requests through STDIN or command line argument and returning responses via STDOUT; something more visual, or even HTML response if you desire.

**Timing Expectations:** You may take as much time as you like on the task, but we don't expect you to spend more than 2-3 hours. If you find you've spent more than 2-3 hours, it is fine to wrap things up and send whatever you have; simply send us a note with the things you'd have liked to have done if you had more time.

**Quality Expectations:** Our only expectation is that the code works to the simple specification provided, and that you are proud of the work product as something you'd be willing to ship. Use the language of your choice, and make use of any external libraries and resources you feel are appropriate.

**Support:** If you have questions about the "specs" then feel free to contact David Block directly at david.block@paymentworks.com.

**Results:** Please complete the task on the following page and submit all code and instructions needed to run the resulting application (including any external libraries that might need to be installed). This may be in any form you'd like that allows us to test and allows us to read the underlying code. You may send your solution via email through your recruiting contact or directly to david.block@paymentworks.com.

**HTTP Data Retrieval and Decoding**

We will be using the API at https://www.mbta.com/developers/v3-api.

1. The two API endpoints we expect you to use are:
    - https://api-v3.mbta.com/routes - lists all the subway lines, commuter-rail lines, bus-lines and ferry-routes
    - https://api-v3.mbta.com/stops - lists all the stops for those routes

2. Your program should allow the user to request a list of **all subway lines** in Boston. The output of this request should provide the user with both the name and "ID" of each line. For example, the output should provide information that includes:
    - ID: Red, NAME: Red Line
    - ID: Green-B NAME: Green Line B
    - ID: Orange NAME: Orange Line
    - ***Note these three bullets are a sample, they are not complete output***

3. Your program should also allow the user to request the list of all stops on a *particular* subway-line, where the user specifies the subway-line by providing that line's "ID". For example, if the user provides the ID "Red", then the resulting output should show all stops on the Red Line.