Hello Ahsan,

Greetings! I hope you're ready for this exercise.

If you have any questions during the test, please contact the following Aeolus engineers, and they'll assist you when they are able to:

- Maciej (maciej.troszynski@aeolusbot.com)
- Samuel (samuel.servulo@aeolusbot.com)

Here is the programming assignment. The spec includes instructions on how to deliver the code. We are targeting 8 total hours of programming, 4 hours for each of the two parts of this task. But we want you to get to a good stopping point on each piece, so while being mindful of the clock, you choose to stop when you are at the right stopping point.
Exercise instructions for second part of coding task I will e-mail you shortly before (November 22, 2017 7:00 am UTC).

Good luck! (and happy hacking!)

OBJECTIVE

Design and implement an internal image processing API, and a command line interface to allow all image operations from the command line. Test, document, and provide implementation notes for your code as you would in a professional setting.

SCOPE

Imagine yourself working as part of a large team, doing a small foundational part of a much larger project. Architect your code such that adding additional functionality, or adding new data representations, will be easy for other members of your team, and can be easily maintained after you've completed the task. This is *very*important.

First, develop a simple set of C++ classes for performing simple image processing tasks which elegantly exposes the image processing operations in a way that might be used within the larger project.

Second, develop a command line interface that allows execution of all operations of the interface from the command line. There are lots of details in the task description below, so be sure to follow these instructions accurately. At the same time, you have a substantial amount of design freedom here to show your skills, as we have deliberately underspecified some tasks. You will deliver your application by checking new code and all associated documentation into Github.

Treat this exercise as if it were a small part of an actual employment engagement: you are building these image processing algorithms to serve as a library within a team, building a suite of applications where both code performance and coding velocity are important. As such your software application will be maintained over time by multiple developers. The goal is to quickly prototype an end-to-end app with the basic functionality specified, then refine your design and implementation as time allows. Develop your code as you would if you were part of our team, with attention to time management, architectural design, code cleanliness, and refactoring.

During this exercise you should pragmatically trade off your time between the normally expected activities including: documenting your APIs, providing test coverage, error handling, refactoring, and code readability.

Your code should have a simple but complete`readme.md` at the root of your github repository which includes all steps needed to compile and execute your command line interface on Ubuntu 16.04. This app must be

coded in C++, it should use openCV **only** to load, save, and display RGB images (you may assume your images are not grayscale), as well as cv::Mat and other basic openCV datatypes.

Please perform very frequent commits as you code, as you would in a real work setting, and do this first half of the assignment in one continuous chunk of time, so we can review your approach. Feel free to refactor, etc. as you go. We hope to give you lots of room to innovate the best design/coding approach and let you do your stuff!

PART I - Create an image analysis service

(1) Implement C++ class(es) to perform "find region", "find perimeter", "display image", and "display pixels" operations:

- **FIND_REGION** accepts an RGB image, an (X, Y) pixel location, and other parameter(s) to control a greedy style algorithm that "flood fills" the nearby pixels in order to find a contiguous patch of pixels that are similar in color. The return value should be some representation of this set of pixels, to be passed to the functions below. Select a reasonable way to control how similar the pixels are; the idea is that regions that a human generally would see as visually similar should be found by your algorithm.
- **FIND_PERIMETER** accepts the output of FIND_REGION and outputs contiguous pixels that border the region. These pixels should be a subset of pixels returned by FIND_REGION.
- **DISPLAY_IMAGE** provides a method for displaying loaded RGB images.
- **DISPLAY_PIXELS** provides a method for displaying an image representation of the results from the above FIND_* operators.
- **SAVE_PIXELS** accepts the output from FIND_* and saves the image to a file.

We understand that developers often make use of online resources such as StackOverflow. While you will be rejected if you copy/paste code someone else's code (just as you could be fired from a real job), using references is okay. Please reference any code sources you employ in algorithm development in your`readme.md` file under a section titled "References".

(2) Implement a command line interface to these functions. The full functionality of each operation should be accessible from the command line, and the interface should support serial application of operations to an initial image. Additionally, the command line tool should be able to write the output data to a text file in some reasonable way.

(3) As mentioned before, place documentation for your class interfaces, and command line tool usage, clearly labeled, in the `readme.md` file. Be sure to explain the meanings of the parameters and the detailed semantics of what your algorithms are trying to achieve (i.e.: what is the meaning and function of the similarity parameter(s) in FIND_REGION).

(4) Test the operations defined above. Use the images and (X, Y) locations identified in the **Test Case folder**that we have provided to test each of the above operations. Optionally, include additional images and CLI example calls that best show off your results from FIND_REGION and FIND_PERIMETER.

Please email me when you have completed this portion of the exercise. Happy hacking!