The following assignment regards an analysis of an image segmentation application, available on the course website as an Eclipse project. The Knox workstations are not equipped with the capabilities needed to run C/C++ Eclipse projects, so it is recommended that you use your own computer, or log on to the Euclid server remotely and use the Eclipse installation there. See the course website for more details regarding Eclipse setup for C/C++ development, in particular for this assignment.

The project contains many files, but in this initial assignment, we need only examine the one called imageSegmentation.cpp. The other included files are left only as resources for future assignments. The project also contains several test pictures. To run the project on the test pictures, you will need to set up a run configuration to provide the executable with command-line arguments for the images. To do so, first try run the application as you normally would, with no command-line arguments (right-click on the project, Run As, Local C/C++ Application.) Once you’ve done this, go to Run, Run Configurations, and under C/C++ Application there should be an entry for ImageSementationg. Go to the Arguments tab, and paste in the following code to get Eclipse to prompt your for arguments when you run the application.

${file_prompt} ${string_prompt:Edge threshold #} ${string_prompt:Minimum object size (in pixels)}

Feel free to experiment with images and segmentation parameters of your own choosing. Some recommended parameters for decent segmentation of the images included are:

- Bricks.jpg 150 1000
- helmet.jpg 250 300
- Nightsky.jpg 200 5
- pottery.jpg 120 300

The application can be broadly divided into four phases – reading the input image from file, finding edges, flood-filling or “coloring” objects based on how the edges divide up the image, and writing out objects in new images to new files. The image reading function is in a separate library (libjpeg,) and is beyond the scope of the project. The remaining three phases are all within the scope of the optimization project we will pursue for the middle third of this course.

Part I: Measuring time within an application (10 points)
The application currently contains a single timer for the total application runtime. Run a baseline performance test on your system, and report the necessary system configuration information that
would be required for someone else to reproduce your result. On a Windows system, for example, you can find out information about your system’s processor and some other components through the system information page under the control panel, also known as the “Properties” page for your computer. Think about what else may be required to report or control, to the best of your ability. Run your experiment a few times to see if the results are stable. Then, measure the runtime required for each of the individual subcomponents, using the method of your choice (modifying the code to time it with the same libraries, using a profiler like Very Sleepy, or any other means that will accomplish the desired goal.) Report the runtimes for each provided test case, and write a paragraph or two describing what the results might mean for someone trying to find the most important part of the application to optimize.

Part II: Locality analysis (9 points)
For each of the three phases you can actually study, describe the access pattern of the primary computational loops, and how it would interact with a typical memory hierarchy in terms of temporal and spatial locality.

Part III: SIMD analysis (9 points)
For the primary loop nests in each of the three analyzable phases, give an initial evaluation on how easily SIMD vectorization could be used. Your descriptions should be short, about a paragraph each, but contain an analysis of memory access strides, control flow within the innermost loop, the presence or absence of loop dependencies, etc.