# **A3 (20 marks)**

***Focus****: OpenMP (B,C,D)*

1. [10 marks] Download *ImageProcessing.zip* file from Canvas. Unzip and copy contents to a new C project. Read the “readme.txt” for copyright info and for some background about the library used.

The first three statement in the main function define the input and output images and the required processing. Play with the values of these constants and check the output.

Your goal is to reduce the processing time by dividing the workload among several threads (e.g., 4 threads) using OpenMP (but don’t use parallel for: i.e., neither #pragma omp parallel for nor #pragma omp for). To verify the correctness of your code, check the output image as well as the processing time (printed out in the console). Report the processing times for 2, 4, 8, 16 threads as a comment in your code.

1. [10 marks] (Adapted from Pacheco-11) Download omp\_trap\_1.c. Run the code (see the note at the end of this assignment). Modify the code twice so that:
2. you perform reduction manually with your own code after the function call. i.e. don’t use the reduction operator; don’t send a pointer for global\_result to Trap function. Instead find the *local* result for each thread and then combine at the end of your main method into a global result (for example, see the lecture notes on Reduction, slide titled: “Area Calculation – Version3”, or theexample shown in the textbook PACHECO-11 page 222).
3. Repeat (a) but with the reduction operator.

For both (a) and (b), compare the execution time of the parallel block using omp\_get\_wtime() which returns the number of seconds that have passed since some time in the past. For details on taking timings, see Section 2.6.4 of the textbook PACHECO-11.

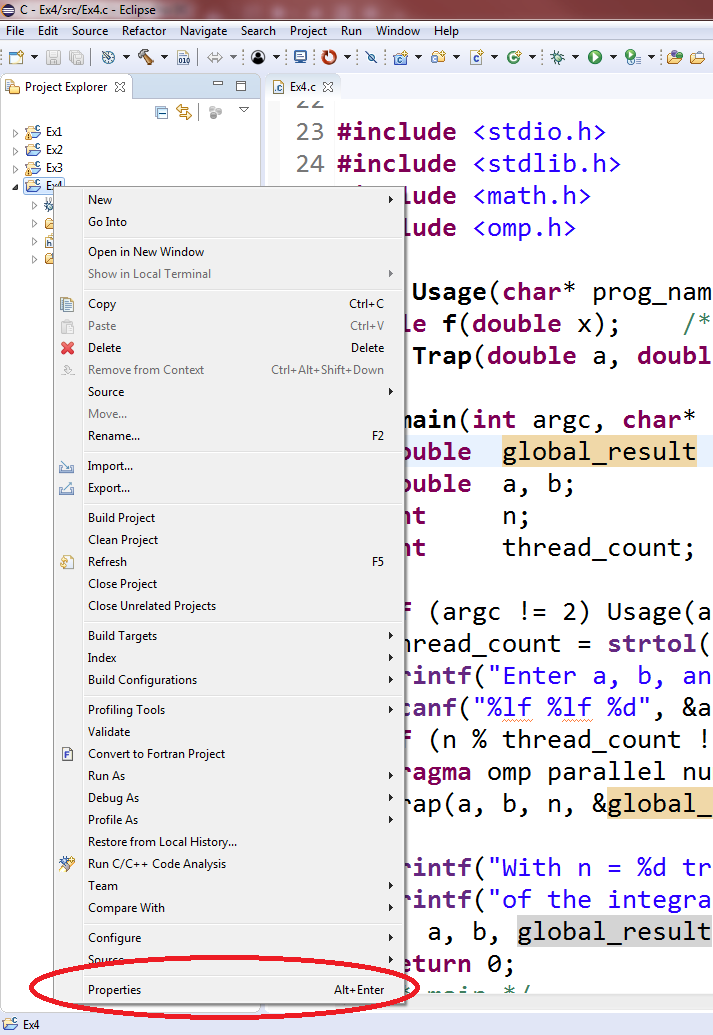
## **Submission Instructions**

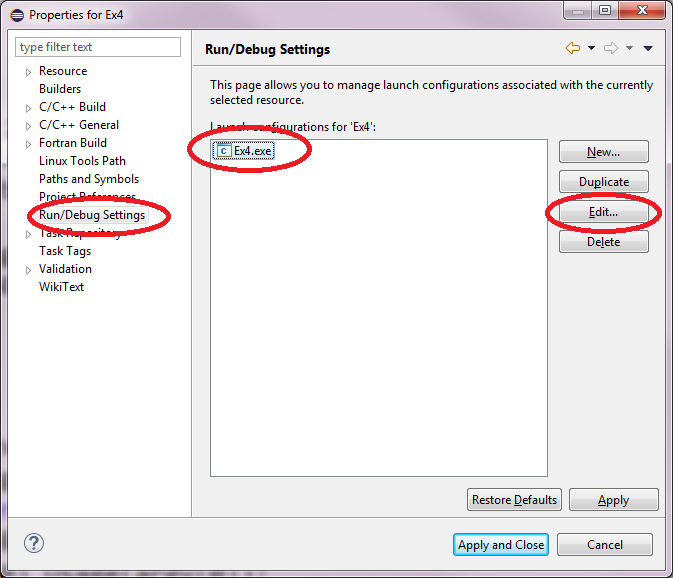
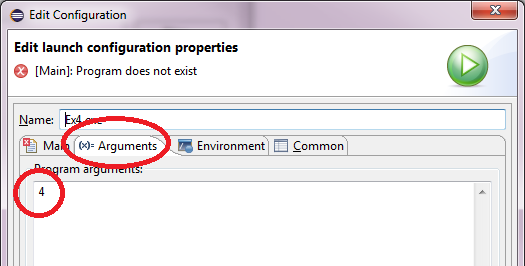
For this assignment, you need to do the following:

1. *Programming questions*: Create one C file for *each* programming question and write your answer inside that file. Your files should have the same name as the question number (e.g., Q1.c)
2. Non-programming questions:
   1. If there are any discussion/essay questions related to a programming question, write your answers as comments at the end of your code for that question.
   2. For all other non-programming questions (i.e., not related to any programming question), write your answers to all of them in *one* Word document file,
3. After solving all questions, compress all your files into one zip folder and give a name to the zipped file that matches your ID (e.g., 1234567.zip).
4. Submit the zipped file **to Canvas.**

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.

**C command line arguments in Eclipse**

The omp\_trap\_1.c program in Q2 above requires passing an argument when running the program (see Usage in the comments part). In order to pass an argument, you can either use command line (i.e. the one in Usage), or you can do this in Eclipse IDE as seen below. Other IDEs usually have a similar way of passing arguments to the execution statement.

1. Right-click the project from the Project Explorer, then choose Properties.
2. Choose Run/Debug Settings. Click your exe file, then Edit…
3. Select the Arguments tab and enter 4 (or another number representing the number of threads) under Program arguments: