**University of Wolverhampton**

**Faculty of Science and Engineering**

**Department of Mathematics and Computer Science**

**Module Assessment**

|  |  |
| --- | --- |
| **Module** | 6CS005 – High Performance Computing |
| **Module Leader** | Hiran Patel |
| **Semester** | 1 |
| **Year** | 2021/22 |
|  |  |
| **Assessment** | Portfolio |
| **% of module mark** | 100% |
| **Due Date** | Date will be published on Canvas |
| **Hand-in – what?** | **Portfolio as specified in this document** |
| **Hand-in- where?** | Canvas |
|  |  |
| **Pass mark** | 40% |
| **Method of retrieval** | Submit the resit assessment (will be distributed at the end of the module) by end of resit week (July) |
| **Feedback** | Individual feedback via Canvas, in addition verbal feedback is available in class. |
| **Collection of marked work** | N/A |

**Assessment overview**

This portfolio is split up into 4 separate tasks which will test your knowledge of advanced multithreading and GPGPU programming using CUDA. Each task should be zipped up into a single zip folder containing all C/CUDA and resource files for the submission on Canvas. All questions below will be explained in week 1 lecture (recorded).

1. Matrix Multiplication (15% - 100 marks)

You will create a matrix multiplication program which uses multithreading, you will be taught two CPU multithreading concepts: POSIX threads and Open Multi-Processing (OpenMP). Matrices are often two-dimensional arrays varying in sizes (can also be 3D), for your application, you will only need to multiply two-dimensional ones. Your program will read in the matrix from a file (txt), store them appropriately using dynamic memory and multiply them by splitting the tasks across “n” threads (any number of threads via command line arguments). Each matrix will be in a separate file. You should also use command line arguments to allow the user to enter the which matrices need to be multiplied. The data files will be given to you however, you will need to initially identify the dimensions of the matrix to determine whether they can be multiplied. If the matrices cannot be multiplied, your program should notify the user. For example, if Matrix A is 3x3 and Matrix B is 2x2, you cannot multiply them. If Matrix A is 2x3 and Matrix B is 2x2, then this can be multiplied. You will need to research how to multiply matrices, this will also be covered in the lectures. The resulting matrix should be outputted to a file. It is up to you which multithreading library you use (Pthreads or OMP).

*You will be given 10 data files to experiment with but remember, the dimensions need to be detected dynamically. During marking, we will be using a different dataset and therefore, do not hard-code your program to work with specific dimensions of matrices.*

**Read data from file appropriately (15 marks)**

**Using dynamic memory (malloc) for matrix A and matrix B (10 marks)**

**Creating an algorithm to multiply matrices correctly (20 marks)**

**Using multithreading with equal computations (35 marks)**

**Printing the correct output matrix to a file (20 marks)**

1. Password cracking using multithreading (15% - 100 marks)

In this task, you will be asked to use the “crypt” library to decrypt a password using multithreading. You will be provided with two programs. The first program called “EncryptSHA512.c” allows you to encrypt a password. For this assessment, you will be required to decrypt a 4-character password consisting of 2 **capital** letters, and 2 numbers. The format of the password should be “LetterLetterNumberNumber.” For example, “HP93.” Once you have generated your password, this should then be entered into your program to decrypt the password. The method of input for the encrypted password is up to you. The second program is a skeleton code to crack the password on a single thread without any multithreading syntax. Your task is to use the pthread or omp library to split the workload over many threads and find the password. Once the password has been found, the program should finish meaning not all combinations of 2 letters and 2 numbers should be explored unless it’s ZZ99 AND the last thread happens to finish last.

**Cracks a password using multithreading and dynamic slicing based on thread count (75 marks)**

**Program finishes appropriately when password has been found (25 marks)**

1. Password Cracking using CUDA (35% - 100 marks)

Using a similar concept as question 2, you will now crack passwords using CUDA. As a kernel function cannot use the crypt library, you will be given an encryption function instead which will generate a password for you. Your program will take in an encrypted password and decrypt it using many threads on the GPU. CUDA allows multidimensional thread configurations so your kernel function (which runs on the GPU) will need to be modified according to how you call your function.

**Generate encrypted password in the kernel function (using CudaCrypt function) to be compared to original encrypted password (25 marks)**

**Allocating the correct amount of memory on the GPU based on input data. Memory is freed once used (15 marks)**

**Program works with varying numbers of blocks and threads (blocks <= 26, threads <= 26) (40 marks)**

**Decrypted password sent back to the CPU and printed (20 marks)**

1. Box Blur using CUDA (35% - 100 marks)

Your program will decode a PNG file into an array and apply the box blur filter. Blurring an image reduces noise by taking the average RGB values around a specific pixel and setting it’s RGB to the mean values you’ve just calculated. This smoothens the colour across a matrix of pixels. For this assessment, you will use a 3x3 matrix. For example, if you have a 5x5 image such as the following (be aware that the coordinate values will depend on how you format your 2D array):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0,4 | 1,4 | 2,4 | 3,4 | 4,4 |
| 0,3 | 1,3 | 2,3 | 3,3 | 4,3 |
| 0,2 | 1,2 | 2,2 | 3,2 | 4,2 |
| 0,1 | 1,1 | 2,1 | 3,1 | 4,1 |
| 0,0 | 1,0 | 2,0 | 3,0 | 4,0 |

The shaded region above represents the pixel we want to blur, in this case, we are focusing on pixel 1,2 (x,y) (Centre of the matrix). to apply the blur for this pixel, you would sum all the **Red** values from the surrounding coordinates including 1,2 (total of 9 R values) and find the average (divide by 9). This is now the new Red value for coordinate 1,2. You must then repeat this for Green and Blue values. This must be repeated throughout the image. If you are working on a pixel which is not fully surrounded by pixels (8 pixels), you must take the average of however many neighbouring pixels there are.

Your task is to use CUDA to blur an image.

NOTE – this program should work with any amount of threads.

**Reading in an image file into a single or 2D array (10 marks)**

**Applying Box filter on image (20 marks)**

**Using multithreading appropriately to apply Box filter (40 marks)**

**Using dynamic memory – malloc (10 marks)**

**Outputting the correct image with Box Blur applied (20 marks)**

**Important Message**

You may be asked to clarify your assessment after moderation has taken place. This is to ensure the work has been completed by the student.

You must achieve 40 percent overall to pass this module. There will be a resit opportunity during resit week (July) to achieve a pass.

**Submission of work**

Your completed work for assignments must be handed in on or before the due date. ***You must keep a copy or backup of any assessed work that you submit.  Failure to do so may result in your having to repeat that piece of work.***

**Penalties for late submission of coursework**

Standard Faculty of Science and Technology arrangements apply.

**ANY late submission (without valid cause) will result in 0 marks being allocated** **to the coursework**.

**Procedure for requesting extensions**

If you have a valid reason for requiring an extension you must request an extension using e:vision. **Requests for extension to assignment deadlines should normally be submitted at least one week before the submission deadline and may be granted for a maximum of seven days (one calendar week).**

**Retrieval of Failure**

A pass of 40% or above must be obtained overall for the module (but not necessarily in each assessment task).

**Where a student fails a module they have the right to attempt the failed assessment(s) once, at the next resit opportunity (normally July resit period).  If a student fails assessment for a second time they have a right to repeat (i.e. RETAKE) the module.**

**NOTE: STUDENTS WHO DO NOT TAKE THEIR RESIT AT THE NEXT AVAILABLE RESIT OPPORTUNITY WILL BE REQUIRED TO REPEAT THE MODULE.**

**Mitigating Circumstances (also called Extenuating Circumstances).**

If you are unable to meet a deadline or attend an examination, and you have a valid reason, then you will need to request via e:vision **Extenuating Circumstances.**

**Feedback of assignments**

You will be given feedback when you demonstrate your work.

You normally have **two working weeks** from the date you receive your grade and feedback to contact and discuss the matter with your lecturer. See the Student’s Union advice page <http://www.wolvesunion.org/adviceandsupport/> for more details.

**Registration**

Please ensure that you are registered on the module. You can check your module registrations via e:Vision You should see your personal tutor or the Student Support Officer if you are unsure about your programme of study. The fact that you are attending module classes does not mean that you are necessarily registered. A grade may not be given if you are not registered.

**Cheating**Cheating is any attempt to gain unfair advantage by dishonest means and includes **plagiarism** and **collusion.** Cheating is a serious offence. You are advised to check the nature of each assessment. You must work individually unless it is a group assessment.

**Cheating** is defined as any attempt by a candidate to gain unfair advantage in an assessment by dishonest means, and includes e.g. all breaches of examination room rules, impersonating another candidate, falsifying data, and obtaining an examination paper in advance of its authorised release.

**Plagiarism** is defined as incorporating a significant amount of un-attributed direct quotation from, or un-attributed substantial paraphrasing of, the work of another.

**Collusion** occurs when two or more students collaborate to produce a piece of work to be submitted (in whole or part) for assessment and the work is presented as the work of one student alone.