6CS005 Learning Journal - Semester 1 2019/20

Srijay Tuladhar 1928952

Table of Contents

1 CUDA

1.1 Password Cracking

Source Code:

```
#include <stdio.h>
#include <cuda_runtime_api.h>
#include <time.h>

/*****************************************************************************

     Compilation Code:
     nvcc -o cuda_password_crack cuda_password_crack.cu

*****************************************************************************/


__device__ int is_a_match(char *attempt) {
  char plain_password1[] = "BV7842";
  char plain_password2[] = "ES2107";
  char plain_password3[] = "HR2332";
  char plain_password4[] = "RB9669";

  char *a = attempt;
  char *b = attempt;
  char *c = attempt;
  char *d = attempt;
  char *p1 = plain_password1;
  char *p2 = plain_password2;
  char *p3 = plain_password3;
  char *p4 = plain_password4;

  while(*a == *p1) {
   if(*a == '\0')
    {
    printf("%s\n",plain_password1);
      break;
    }

    a++;
    p1++;
  }

  while(*b == *p2) {
   if(*b == '\0')
    {
    printf("%s\n",plain_password2);
      break;
}

    b++;
    p2++;
  }

  while(*c == *p3) {
   if(*c == '\0')
    {
    printf("%s\n",plain_password3);
      break;
    }

    c++;
    p3++;
  }

  while(*d == *p4) {
   if(*d == '\0')
    {
    printf("%s",plain_password4);
      return 1;
    }

    d++;
    p4++;
  }
  return 0;

}
/****************************************************************************
  The kernel function assume that there will be only one thread and uses
  nested loops to generate all possible passwords and test whether they match
  the hidden password.
*****************************************************************************/

__global__ void  kernel() {
char k1,k2,k3,k4;

  char password[7];
  password[6] = '\0';

```

```
int i = blockIdx.x+65;
int j = threadIdx.x+65;
char firstValue = i;
char secondValue = j;

password[0] = firstValue;
password[1] = secondValue;
    for(k1='0'; k1<='9'; k1++){
      for(k2='0'; k2<='9'; k2++){
        for(k3='0'; k3<='9'; k3++){
          for(k4='0'; k4<='9'; k4++){
            password[2] = k1;
            password[3] = k2;
            password[4] = k3;
                   password[5] = k4;
                 if(is_a_match(password)) {
               //printf("Success");
                 }
                    else {
                //printf("tried: %s\n", password);
                   }
                 }
               }
             }
          }
       }
       int time_difference(struct timespec *start,
                           struct timespec *finish,
                           long long int *difference) {
         long long int ds =  finish->tv_sec - start->tv_sec;
         long long int dn =  finish->tv_nsec - start->tv_nsec;

         if(dn < 0 ) {
           ds--;
           dn += 1000000000;
         }
         *difference = ds * 1000000000 + dn;
         return !(*difference > 0);
       }


       int main() {

         struct  timespec start, finish;
         long long int time_elapsed;
         clock_gettime(CLOCK_MONOTONIC, &start);
         printf("\n===============================================================================\n");
```

```c
    printf("!! MATCHED PASSWORD !! \n");
    printf("================================================================================\n\n");
    kernel <<<26,26>>>();
    cudaThreadSynchronize();


    clock_gettime(CLOCK_MONOTONIC, &finish);
    time_difference(&start, &finish, &time_elapsed);
    printf("\n\n================================================================================\n");
    printf("!! TIME TAKEN FOR EXECUTION !! \n");
    printf("================================================================================\n\n");
    printf("Nanoseconds: %lld\n", time_elapsed);
    printf("Seconds: %0.9lf\n", ((time_elapsed/1.0e9)));
    printf("Minutes: %0.4lf\n", ((time_elapsed/1.0e9)/60));
    printf("Hours: %0.2lf\n\n", ((time_elapsed/1.0e9)/3600));


    return 0;
  }
```

Insert a table that shows running times for the original and CUDA versions.

Attempt	Time taken for Execution (in minutes)	
	Original Program	CUDA Version
1	8.0926	0.0022
2	8.6583	0.0020
3	7.9492	0.0020
4	7.9292	0.0021
5	7.9518	0.0019
6	8.1267	0.0020
7	7.9703	0.0021
8	8.0024	0.0021
9	7.9405	0.0020
10	8.1816	0.0021
Mean Running Time	**8.08026**	**0.00205**

Write a short analysis of the results

Analysis:

As from the above table of comparison between the original program and the CUDA version of the program, there is drastic difference in the mean running time, where the time taken for execution for the CUDA version is almost 4000 times faster than that of the original version. The thread used in GPU is 26 in a single block where the thread executed is 26 times 26, because of which there is vast difference in the result.

1.2 Image Processing

Source Code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <malloc.h>
#include <signal.h>
#include <cuda_runtime_api.h>

/******************************************************************************

   Compilation Code:
   nvcc -o cuda_image_processing cuda_image_processing.cu -lglut -lGL -lm

******************************************************************************/

#define width 100
#define height 72

unsigned char image[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 };

 unsigned char results[width * height];
 static void key_pressed(unsigned char key, int x, int y);
 void stgint callback(int signal_number);
 static void display();
 void tidy_and_exit();

 __global__ void detect_edges(unsigned char *in, unsigned char *out) {

   unsigned int i = blockIdx.x ;

     int x;
     int y;
     int b;
     int d;
     int f;
     int h;
```

```
          int r;

          y = i / 100;
          x = i - (100 * y);

          if (x == 0 || y == 0 || x == width - 1 || y == height - 1) {
            out[i] = 0;
          } else {
            b = i + 100;
            d = i - 1;
            f = i + 1;
            h = i - 100;

            r = (in[i] * 4) + (in[b] * -1) + (in[d] * -1) + (in[f] * -1) + (in[h] * -1);

            if (r > 0) { // if the result is positive this is an edge pixel
              out[i] = 255;
            } else {
              out[i] = 0;
            }
          }
        }




      void tidy_and_exit() {
        exit(0);
      }

      void sigint_callback(int signal_number){
        printf("\nInterrupt from keyboard\n");
        tidy_and_exit();
      }

      static void display() {
        glClear(GL_COLOR_BUFFER_BIT);
        glRasterPos4i(-1, -1, 0, 1);
        glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, image);
        glRasterPos4i(0, -1, 0, 1);
        glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, results);
        glFlush();
      }

      static void key_pressed(unsigned char key, int x, int y) {
        switch(key){
          case 27: // escape
```

```
              tidy_and_exit();
              break;
            default:
              printf("\nPress escape to exit\n");
              break;
          }
        }
        int time_difference(struct timespec *start, struct timespec *finish,
                                      long long int *difference) {
          long long int ds =  finish->tv_sec - start->tv_sec;
          long long int dn =  finish->tv_nsec - start->tv_nsec;

          if(dn < 0 ) {
            ds--;
            dn += 1000000000;
          }
          *difference = ds * 1000000000 + dn;
          return !(*difference > 0);
        }

        int main(int argc, char **argv) {


          signal(SIGINT, sigint_callback);

          printf("image dimensions %dx%d\n", width, height);


          unsigned char *d_results;
          unsigned char *d_image;

          cudaMalloc((void**)&d_results, sizeof(unsigned char) * (width * height));
          cudaMalloc((void**)&d_image, sizeof(unsigned char) * (width * height) );
          cudaMemcpy(d_image, &image, sizeof(unsigned char) * (width * height), cudaMemcpyHostToDevice);
          cudaMemcpy(&d_results, &results, sizeof(unsigned char) * (width * height), cudaMemcpyHostToDevice);

          struct timespec start, finish;
          long long int time_elapsed;

          clock_gettime(CLOCK_MONOTONIC, &start);


          printf("\n===============================================================================\n");
          printf("!! IMAGE PROCESSING  !! \n");
          printf("===============================================================================\n\n");

          detect_edges <<<7200, 1>>>(d_image, d_results);
          cudaThreadSynchronize();
          clock_gettime(CLOCK_MONOTONIC, &finish);

          time_difference(&start, &finish, &time_elapsed);

          printf("\n\n===============================================================================\n");
          printf("!! TIME TAKEN FOR EXECUTION !! \n");
          printf("===============================================================================\n\n");
          printf("Nanoseconds: %lld\n", time_elapsed);
          printf("Seconds: %0.9lf\n", ((time_elapsed/1.0e9)));
          printf("Minutes: %0.4lf\n", ((time_elapsed/1.0e9)/60));
          printf("Hours: %0.2lf\n\n", ((time_elapsed/1.0e9)/3600));

          cudaMemcpy(&results, d_results, sizeof(unsigned char) * (width * height), cudaMemcpyDeviceToHost);
          cudaMemcpy(&image, &d_image, sizeof(unsigned char) * (width * height), cudaMemcpyDeviceToHost);

          cudaFree(&d_image);
          cudaFree(&d_results);

          glutInit(&argc, argv);
          glutInitWindowSize(width * 2,height);
          glutInitDisplayMode(GLUT_SINGLE | GLUT_LUMINANCE);

          glutCreateWindow("6CS005 Image Progessing Courework");
          glutDisplayFunc(display);
          glutKeyboardFunc(key_pressed);
          glClearColor(0.0, 1.0, 0.0, 1.0);

          glutMainLoop();
          tidy_and_exit();


          return 0;
        }
```

Insert a table that shows running times for the original and CUDA versions.

Attempt	Time taken for Execution (in seconds)	
	Original Program	CUDA Version
1	0.000095213	0.000007852
2	0.000173475	0.000012254
3	0.000099872	0.000011081
4	0.000092141	0.000019027
5	0.000097617	0.000011382
6	0.000104226	0.000008241
7	0.000089945	0.000009571
8	0.000094815	0.000011360
9	0.000090229	0.000014032
10	0.000095256	0.000056569
Mean Running Time	**0.000103279**	**0.0000161369**

Write a short analysis of the results

<u>Analysis:</u>

Observing the above table, the difference in mean running time of the original version and CUDA version of the program is displayed. By calculating the difference in mean running time between both versions, the CUDA version is 6.4 times faster than the original version of the program. The thread used is 7200 for a single block in the GPU because of which the CUDA version of the program is executed faster.

1.3 Linear Regression

Source Code:

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include <cuda_runtime_api.h>
#include <errno.h>
#include <unistd.h>
/******************************************************************************

    Compilation Code:
    nvcc -o cuda_linear_regression cuda_linear_regression.cu -lm

******************************************************************************/

typedef struct point_t{
double x;
double y;
}point_t;

int n_data = 1000;
__device__ int d_n_data = 1000;

point_t data[] = {
  {82.45,155.07},{65.27,121.45},{67.29,117.39},{72.17,120.95},
  {69.42,132.58},{76.35,137.91},{79.20,151.69},{71.97,123.08},
  {85.03,137.12},{78.83,136.47},{71.34,131.75},{66.14,129.76},
  {65.22,111.73},{77.67,137.24},{73.30,105.03},{71.56,120.18},
  {66.92,105.91},{69.09,134.67},{54.03,108.08},{61.79,114.62},
  {67.52,119.60},{31.12,75.51},{13.49,50.66},{61.43,134.15},
  {51.51,107.20},{93.87,149.32},{98.59,167.92},{94.93,146.15},
  {32.47,67.59},{36.91,92.19},{45.36,104.11},{42.58,97.37},
  { 2.38,35.79},{52.07,114.35},{40.76,111.33},{35.44,98.07},
  {57.03,114.02},{17.15,65.52},{26.63,75.12},{68.64,132.38},
  {87.73,137.17},{43.40,106.42},{59.12,103.58},{ 5.83,35.24},
  {31.03,79.78},{68.56,127.27},{21.54,60.20},{19.62,67.80},
  {61.39,128.09},{45.79,89.44},{16.02,64.22},{19.78,65.61},
  {34.76,88.37},{45.97,85.20},{88.74,145.02},{76.48,129.69},
  {19.76,56.76},{87.72,157.39},{66.75,118.41},{63.57,121.44},
  {29.80,87.78},{32.63,85.94},{75.87,134.69},{ 0.85,40.28},
  {94.47,163.58},{72.99,135.55},{64.22,127.04},{ 3.32,40.20},
  { 6.88,42.32},{31.08,75.99},{60.22,120.13},{17.45,60.36},
  {57.29,105.03},{49.31,82.69},{11.87,61.21},{81.39,144.96},
  {48.71,78.63},{11.23,36.32},{16.35,54.14},{19.70,57.20},
  {17.03,63.54},{84.59,154.43},{ 7.41,42.73},{43.82,81.77},
  {49.21,107.45},{53.00,95.15},{13.27,45.40},{67.77,128.98},
  {93.90,165.29},{93.29,173.90},{40.02,86.36},{22.79,75.44},
  {98.39,167.10},{94.95,164.10},{60.08,127.24},{ 6.06,45.37},
  {61.35,121.18},{ 4.95,27.34},{23.32,66.61},{32.79,88.38},
  {83.20,143.97},{60.59,126.92},{ 7.93,70.52},{94.46,152.96},
  {71.82,121.20},{59.27,107.93},{64.93,142.11},{94.58,163.94},
  { 7.21,40.58},{52.74,116.16},{79.42,120.53},{ 7.82,30.57},
  { 9.83,45.29},{58.21,108.86},{48.11,107.39},{88.55,140.09},
  {29.26,71.16},{34.96,80.00},{ 1.12,23.12},{55.14,104.13},
  {19.82,50.32},{43.38,83.62},{24.62,51.03},{62.84,101.88},
  {26.88,60.06},{94.48,144.08},{95.14,157.80},{47.90,85.47},
  {90.90,143.82},{39.53,79.74},{80.77,155.44},{ 6.07,17.43},
  {56.88,103.70},{43.95,84.71},{16.12,45.98},{ 5.12,44.40},
  {81.71,134.56},{24.30,45.54},{83.68,146.90},{17.62,49.28},
  {42.10,97.75},{41.25,84.38},{82.68,155.74},{44.56,95.45},
  {85.21,142.50},{73.50,125.45},{ 3.45,52.95},{30.65,73.60},
  {29.33,76.20},{30.31,85.46},{69.41,135.79},{73.21,133.16},
  {40.62,87.68},{26.38,65.16},{ 5.14,59.66},{94.33,160.01},
  { 6.52,52.57},{90.79,146.06},{ 9.78,55.77},{ 4.71,53.43},
  {74.01,129.97},{68.72,119.11},{16.35,59.99},{44.08,109.17},
  {31.02,63.78},{14.76,33.17},{62.63,126.09},{55.88,96.90},
  {57.41,99.30},{83.66,131.04},{86.08,175.22},{81.13,140.01},
  {18.25,71.09},{65.68,104.02},{66.08,122.24},{48.81,96.28},
  {79.07,132.27},{20.07,67.34},{16.24,48.49},{30.98,85.11},
  { 2.27,45.14},{44.11,76.86},{ 2.49,45.65},{72.96,136.23},
  {89.49,156.60},{54.51,105.71},{92.23,153.22},{95.02,160.48},
  {73.99,111.16},{52.70,93.18},{90.82,154.82},{53.42,100.57},
  {19.77,60.95},{26.30,63.93},{23.07,54.59},{88.86,142.32},
  {98.65,175.75},{76.19,130.10},{59.20,111.38},{58.43,121.18},
  {33.27,82.74},{74.68,126.95},{88.64,141.44},{81.47,117.66},
  {99.22,170.99},{98.17,163.34},{91.54,144.52},{17.22,67.20},
  {66.49,115.36},{68.68,128.45},{ 1.35,54.22},{47.22,98.90},
  {79.94,147.19},{22.05,76.35},{50.23,102.66},{ 5.97,37.93},
  {67.56,98.13},{18.19,52.11},{81.03,149.27},{45.50,98.92},
  {50.60,91.80},{73.59,129.07},{88.92,139.84},{92.80,159.34},
  { 6.39,45.68},{64.04,109.08},{57.32,111.22},{36.89,82.67},
  { 2.04,47.08},{ 3.58,43.67},{66.42,131.32},{81.67,145.83},
  { 3.01,28.87},{30.05,69.62},{32.51,91.29},{32.10,56.40},
  {74.96,121.89},{66.82,125.73},{72.51,129.45},{ 5.91,48.37},
  {37.12,82.47},{ 9.16,48.40},{13.04,46.47},{48.80,95.11},
  {58.51,112.16},{44.86,85.77},{56.11,123.07},{82.96,151.82},
  {24.90,79.21},{27.30,64.03},{99.30,144.46},{62.24,117.56},
  {52.10,91.97},{39.86,79.58},{15.84,72.42},{91.38,151.59},
  {39.75,76.49},{49.68,92.98},{53.69,123.67},{76.59,145.25},
  {84.40,156.17},{81.04,142.59},{24.22,48.48},{63.39,115.54},
  {10.21,40.70},{41.56,62.95},{88.85,137.60},{50.03,118.66},
  {48.66,89.36},{57.74,104.91},{74.07,144.74},{77.68,138.69},
  {98.53,163.18},{25.40,89.65},{ 4.38,50.45},{59.86,102.93},
  { 2.27,42.85},{81.03,143.24},{20.95,76.89},{52.59,116.92},
  {82.19,145.87},{51.90,110.85},{43.83,105.20},{44.13,75.17},
  {17.22,61.38},{46.16,92.95},{55.00,117.41},{ 7.73,39.87},
  {95.80,164.28},{59.80,104.95},{22.16,52.76},{82.10,141.69},
  {94.60,160.59},{18.61,28.99},{ 0.09,47.91},{91.39,158.91},
  {65.15,130.03},{ 7.51,53.66},{64.79,130.85},{15.19,69.90},
  {44.93,89.05},{18.02,63.77},{18.65,61.04},{66.05,134.15},
         {41.95,77.11},{71.75,132.82},{86.89,161.83},{40.11,80.13},
         {11.56,54.38},{15.36,72.22},{38.06,89.41},{99.49,182.71},
         {11.80,44.98},{32.91,77.44},{92.77,151.86},{16.94,68.22},
         {17.24,56.67},{68.12,142.77},{68.15,127.99},{ 3.56,36.04},
         {53.17,102.91},{59.10,107.60},{16.95,58.11},{61.04,116.90},
         {67.28,132.10},{34.20,67.56},{70.29,130.78},{75.05,117.15},
         {96.04,161.15},{16.32,46.04},{ 7.14,43.90},{96.30,167.24},
         {99.45,167.72},{15.83,47.52},{74.86,114.53},{37.08,96.05},
         { 6.63,31.29},{76.68,140.83},{38.03,89.69},{35.38,82.67},
         {99.18,136.72},{ 1.49,35.32},{40.86,71.52},{36.16,87.19},
         {46.66,109.91},{89.29,167.46},{55.40,97.42},{34.92,95.51},
         {30.80,86.35},{25.23,63.36},{46.36,86.14},{13.89,65.48},
         {55.55,93.72},{25.25,51.43},{82.79,139.96},{52.15,101.20},
         {31.66,66.89},{43.96,83.82},{15.40,61.96},{97.62,161.90},
         {17.03,44.60},{53.29,93.54},{64.91,130.41},{73.78,142.21},
         {59.51,107.07},{87.11,153.09},{86.41,161.30},{17.11,70.42},
         {15.93,70.49},{54.23,109.78},{62.93,109.82},{34.17,82.60},
         {68.34,146.39},{28.41,64.48},{76.80,129.30},{95.42,151.63},
         {64.32,116.92},{93.89,159.68},{74.96,149.71},{14.27,46.96},
         {10.64,50.39},{17.18,43.97},{ 2.92,52.04},{96.04,167.13},
         {48.51,101.01},{36.54,74.86},{35.91,75.86},{74.21,132.27},
         {99.87,149.79},{82.35,148.39},{51.71,103.93},{74.97,133.12},
         {94.46,157.28},{34.36,78.95},{40.30,92.46},{99.73,167.41},
         {52.16,108.47},{58.01,102.16},{96.05,145.45},{17.18,54.94},
         { 2.62,40.96},{30.13,65.42},{13.35,58.22},{71.31,125.60},
         {95.70,158.35},{ 2.73,45.15},{97.83,179.16},{28.52,71.03},
         {65.27,103.35},{77.65,126.47},{44.02,99.96},{31.50,71.98},
         {30.92,68.42},{ 3.90,33.31},{81.52,133.74},{64.99,132.19},
         { 7.06,55.22},{71.10,128.30},{43.63,88.87},{14.62,60.91},
         {57.96,102.69},{22.60,74.92},{71.02,120.52},{72.80,136.35},
         {79.02,126.69},{52.49,112.59},{ 0.19,47.94},{47.95,94.10},
         {10.43,52.00},{57.04,124.36},{94.75,176.85},{ 6.21,50.17},
         {77.08,136.86},{38.25,98.59},{96.31,153.49},{15.63,50.58},
         {48.07,96.65},{29.37,91.68},{93.95,162.29},{14.86,64.86},
         {55.48,117.13},{39.49,78.66},{17.29,63.56},{21.38,54.13},
         {67.63,124.02},{18.74,47.72},{70.95,110.97},{63.18,120.04},
         {82.09,145.44},{79.27,140.28},{23.30,75.42},{58.07,128.54},
         { 1.17,38.14},{43.35,85.94},{70.04,125.53},{93.60,159.75},
         { 9.74,42.74},{66.15,119.70},{99.91,153.79},{86.24,170.84},
         {70.67,138.70},{49.61,110.31},{17.22,70.28},{46.41,98.86},
         {19.76,65.18},{71.78,151.92},{88.22,158.34},{20.27,53.32},
         { 6.66,38.32},{82.44,145.08},{75.28,135.37},{17.33,69.56},
         {25.39,90.00},{99.22,175.85},{45.15,86.49},{98.20,166.92},
         {68.65,115.71},{91.06,150.84},{88.26,153.55},{ 4.07,47.73},
         {35.18,84.76},{ 1.72,49.59},{13.84,69.71},{32.88,64.06},
         {28.82,79.54},{14.98,60.96},{91.34,147.91},{94.29,153.25},
         {39.27,91.57},{99.21,173.80},{15.22,59.83},{37.42,94.80},
         {23.35,49.48},{56.46,91.68},{79.14,148.27},{13.71,62.49},
         {45.44,92.67},{27.76,65.51},{72.71,127.57},{79.76,138.44},
         {67.54,100.64},{44.33,92.14},{19.99,54.33},{13.21,59.86},
         {82.42,137.42},{56.86,101.23},{18.29,44.21},{83.90,126.19},
         {54.32,117.82},{11.57,59.56},{40.22,90.54},{ 0.97,24.21},
         {13.29,55.09},{61.92,105.11},{19.82,81.97},{57.73,96.16},
         {38.86,89.80},{86.58,153.61},{62.66,121.44},{85.51,134.84},
         {91.57,158.71},{ 8.84,49.59},{91.57,136.11},{39.01,90.65},
         {41.64,88.50},{77.06,146.16},{41.58,96.92},{29.78,72.24},
         { 9.31,63.47},{ 4.12,44.88},{85.92,150.99},{90.09,151.84},
         {46.27,95.59},{84.84,134.93},{26.34,57.57},{50.43,96.16},
         { 2.88,25.83},{ 7.11,50.96},{16.51,47.60},{73.89,114.11},
         {45.32,88.11},{88.84,132.51},{80.00,123.54},{ 6.47,47.79},
         {60.00,106.47},{75.72,146.29},{10.65,62.48},{31.23,73.26},
         {77.53,121.10},{40.60,95.22},{48.72,94.30},{50.23,88.26},
         {96.85,159.63},{57.33,125.40},{64.74,129.05},{24.94,61.85},
         {82.47,147.83},{67.22,124.22},{76.66,131.25},{73.56,151.75},
         {19.36,56.66},{83.01,115.34},{41.98,79.77},{27.09,65.30},
         {90.54,141.86},{81.78,137.00},{53.45,80.21},{84.43,145.49},
         {34.04,84.18},{64.75,142.10},{60.98,106.50},{87.76,147.41},
         {77.76,138.39},{80.04,145.45},{26.05,94.32},{97.00,170.04},
         {42.05,98.36},{21.13,70.60},{29.70,67.99},{33.38,61.69},
         {50.16,89.72},{50.22,100.23},{63.60,120.36},{13.76,54.38},
         {53.43,110.84},{71.37,144.37},{ 8.10,56.51},{50.47,119.27},
         {50.65,96.47},{10.14,49.66},{ 7.79,74.00},{67.56,119.06},
         {58.93,113.17},{24.89,41.82},{52.45,102.32},{32.08,64.43},
         {11.02,57.50},{94.14,164.65},{75.71,127.33},{83.84,134.81},
         {96.60,168.54},{72.00,135.66},{53.03,105.83},{32.21,58.94},
         {31.03,79.56},{83.04,144.26},{78.58,137.20},{87.36,140.76},
         {68.41,150.16},{ 8.12,54.89},{63.22,118.29},{27.54,63.52},
         {53.60,100.09},{60.42,98.19},{ 6.88,55.69},{26.33,69.75},
         {72.19,132.73},{70.87,125.99},{97.80,168.70},{47.03,88.44},
         {18.91,84.53},{10.86,56.49},{95.26,166.77},{89.35,160.12},
         { 1.11,29.40},{71.91,124.64},{50.05,92.00},{ 1.88,49.75},
         {33.74,75.65},{99.84,164.44},{17.57,53.77},{75.64,137.60},
         { 6.76,38.31},{15.42,54.80},{90.43,151.35},{38.00,86.86},
         {54.83,128.48},{ 5.00,48.26},{99.41,165.03},{55.49,136.74},
         {17.69,66.98},{78.11,165.26},{74.17,117.71},{52.17,95.12},
         {33.65,89.10},{31.03,88.57},{76.86,117.08},{96.81,165.16},
         {21.64,75.28},{86.85,145.70},{85.75,158.93},{29.87,74.72},
         {11.91,44.00},{23.40,74.94},{88.53,148.97},{70.23,124.86},
         {43.71,91.50},{49.77,85.70},{29.28,67.78},{12.04,53.16},
         {54.39,92.06},{51.96,85.72},{69.06,128.88},{80.24,150.69},
         {26.16,69.57},{60.24,134.05},{ 3.23,34.58},{43.07,111.18},
         { 8.28,46.68},{23.92,56.04},{50.95,80.65},{17.20,40.50},
         {55.76,107.63},{ 2.94,55.66},{80.80,152.89},{72.09,129.29},
         {23.06,46.95},{54.25,118.47},{74.87,129.45},{18.46,52.04},
         {46.08,98.46},{15.14,43.60},{75.59,119.50},{ 8.46,26.29},
         {38.03,67.55},{20.59,80.62},{42.95,99.22},{14.76,48.50},
         {62.18,107.07},{ 2.41,46.26},{68.55,139.84},{91.19,156.14},
         {65.64,153.56},{26.91,67.76},{84.73,141.90},{55.04,114.08},
         {53.28,96.66},{72.34,121.86},{35.21,61.10},{25.86,68.32},
         {40.80,70.62},{83.16,136.63},{ 1.84,44.66},{98.14,165.56},
         {92.78,166.98},{ 4.08,41.70},{ 1.70,32.25},{24.23,63.25},
         {72.69,139.53},{11.85,54.34},{17.17,64.66},{34.42,71.95},
         {48.25,109.59},{41.39,85.48},{ 3.11,51.08},{98.52,174.32},
         {64.12,116.37},{21.65,72.72},{69.95,142.06},{85.71,138.26},
         {74.60,133.55},{18.65,49.50},{12.47,43.50},{85.34,142.94},
         {54.57,116.95},{37.47,87.34},{81.35,156.19},{90.42,167.55},
         {32.62,83.33},{43.90,81.36},{40.76,83.87},{27.46,61.84},
         { 0.71,39.30},{50.49,97.46},{63.21,104.66},{85.29,143.18},
         {66.07,118.09},{41.01,62.63},{70.07,107.34},{89.88,146.24},
         {24.27,72.41},{11.67,52.46},{ 2.46,45.31},{90.44,152.17},
         {30.21,63.25},{19.93,51.17},{54.78,103.51},{81.78,137.70},
         {50.42,95.37},{36.57,84.66},{56.07,99.49},{93.33,171.32},
         {42.89,81.41},{95.73,146.55},{15.09,48.90},{38.77,77.29},
         {25.12,72.50},{51.68,116.94},{73.35,131.87},{86.30,141.22},
         {18.64,68.35},{42.82,103.58},{18.05,60.95},{ 0.93,42.06},
         {51.92,105.51},{86.17,151.87},{78.51,132.91},{71.60,138.14},
         {60.94,107.61},{25.73,73.76},{89.77,146.34},{17.86,66.42},
         {17.32,62.95},{17.74,58.61},{17.62,74.78},{29.49,69.46},
         { 6.97,46.16},{66.82,122.03},{65.83,125.74},{81.11,141.75},
         { 3.66,41.01},{47.10,103.63},{30.08,92.55},{13.74,57.80},
         {71.11,119.96},{85.53,134.01},{30.06,75.18},{ 6.39,55.28},
         { 4.71,58.24},{90.58,156.30},{33.88,74.17},{30.15,58.67},
         { 3.13,45.77},{48.51,92.11},{32.87,80.67},{23.06,83.17},
         {15.07,56.49},{22.75,76.55},{65.04,133.02},{66.48,107.61},
         {10.28,49.68},{59.05,107.49},{19.16,67.00},{60.15,101.76},
         {65.10,114.80},{76.70,132.78},{38.18,81.59},{22.45,71.10},
         { 5.95,48.36},{10.36,56.33},{21.70,67.53},{89.43,150.56},
         {90.66,145.45},{18.83,66.13},{37.02,81.86},{83.30,136.05},
         {49.76,96.94},{ 8.59,42.07},{99.14,165.45},{66.61,140.27},
         {59.13,106.74},{13.69,64.66},{ 3.69,37.62},{82.55,152.57},
         {16.86,59.16},{45.19,105.01},{93.84,162.69},{21.89,86.05},
         {61.30,108.80},{41.07,89.96},{49.43,89.37},{72.23,122.68},
         {30.12,62.82},{ 3.66,51.65},{92.08,146.13},{14.08,51.36},
         {70.36,109.49},{49.30,95.77},{30.97,86.91},{37.02,86.69},
         {87.33,159.73},{ 9.21,50.78},{56.33,97.30},{87.10,151.05},
         {96.46,176.35},{32.08,79.44},{39.92,78.08},{34.26,71.62},
         {54.20,116.50},{61.93,143.59},{ 0.17,28.98},{20.02,68.47},
         {67.10,124.67},{10.50,55.32},{17.92,80.62},{ 1.13,49.11},
         {23.42,61.62},{20.61,60.61},{58.59,130.42},{45.68,109.39},
         {40.65,89.41},{40.52,96.88},{32.28,98.28},{24.68,70.29},
         {97.32,146.42},{ 6.22,68.36},{64.16,112.26},{58.26,100.94},
         {52.43,102.08},{35.20,91.98},{99.87,169.63},{ 7.17,41.08},
         {92.21,152.49},{89.21,163.34},{94.95,160.36},{ 6.20,52.92},
         {24.68,69.97},{88.56,166.68},{24.08,74.85},{20.38,66.00},
         {84.57,148.39},{84.11,139.97},{40.21,105.66},{51.88,84.25},
         {19.02,75.66},{97.92,164.22},{38.86,100.02},{76.97,131.01},
         {85.08,145.73},{55.31,110.56},{58.80,123.03},{30.48,68.51},
         {90.37,161.69},{92.93,157.06},{62.33,111.57},{28.72,67.78},
         {66.38,117.51},{74.84,125.32},{62.34,127.23},{93.96,149.34},
         {70.54,128.38},{78.01,139.64},{47.93,102.30},{61.76,122.96},
         {88.68,152.56},{26.34,61.63},{50.17,104.98},{17.34,59.56},
         {50.20,99.25},{24.46,71.96},{22.46,44.42},{75.85,118.58},
         {22.97,77.21},{85.67,161.32},{32.35,98.54},{15.42,45.56},
         {41.59,77.31},{82.11,143.74},{54.00,113.73},{ 3.46,59.65},
         { 1.92,34.47},{32.21,82.73},{39.94,78.28},{25.55,48.17},
         { 7.17,36.43},{ 8.83,24.42},{84.19,130.80},{10.86,54.87},
         {44.58,86.79},{30.70,84.62},{ 2.96,44.81},{68.91,124.92},
         { 3.96,46.02},{ 9.65,33.46},{12.03,57.22},{50.41,96.71},
         {17.40,61.16},{69.93,128.22},{93.95,147.08},{16.05,60.44},
         {31.23,91.22},{51.78,91.57},{77.23,138.76},{14.60,60.31},
         {58.51,105.52},{27.08,63.96},{95.07,163.48},{29.52,74.84},
         {63.46,117.37},{82.11,139.92},{76.64,137.90},{28.58,74.39},
         {19.20,62.95},{60.15,125.63},{99.02,157.54},{73.31,117.87},
         {92.20,153.13},{90.70,154.11},{ 5.70,47.08},{60.30,108.19},
         {32.09,70.53},{28.52,63.25},{10.76,49.56},{ 2.35,37.68},
         {57.60,100.04},{26.49,66.68},{93.57,167.30},{25.95,85.51},
         { 7.44,39.17},{58.98,118.56},{21.96,58.41},{12.65,46.49},
         {25.43,61.37},{17.02,49.31},{98.97,176.85},{45.53,83.28},
         {65.89,127.86},{49.86,99.94},{16.78,57.64},{95.62,151.48},
         {24.37,48.55},{57.74,113.98},{26.07,78.93},{14.95,71.57},
         {28.77,66.55},{15.07,43.63},{80.59,137.39},{64.30,128.21},
         {81.54,107.43},{86.39,160.85},{87.96,138.03},{35.68,95.12},
         {17.28,55.07},{90.78,154.10},{88.52,163.38},{92.19,163.85},
         {61.82,119.93},{52.13,107.98},{89.66,142.94},{94.27,166.71}
       };
       double residual_error(double x, double y, double m, double c) {
         double e = (m * x) + c - y;
         return e * e;
       }
       __device__ double d_residual_error(double x, double y, double m, double c) {
         double e = (m * x) + c - y;
         return e * e;
       }
       double rms_error(double m, double c) {
         int i;
         double mean;
         double error_sum = 0;

         for(i=0; i<n_data; i++) {
           error_sum += residual_error(data[i].x, data[i].y, m, c);
         }

         mean = error_sum / n_data;

         return sqrt(mean);
       }
       __global__ void d_rms_error(double *m, double *c,double *error_sum_arr,point_t *d_data) {
         int i = threadIdx.x + blockIdx.x *blockDim.x;
       error_sum_arr[i] = d_residual_error(d_data[i].x,d_data[i].y, *m, *c);
       }

       int time_difference(struct timespec *start, struct timespec *finish, long long int *difference)
       {
       long long int ds = finish->tv_sec - start->tv_sec;
       long long int dn = finish->tv_nsec - start->tv_nsec;

        if(dn < 0){
         ds--;
         dn += 1000000000;
       }
         *difference = ds * 1000000000 + dn;
         return !(*difference > 0);
       }



       int main(){
        int i;
         double bm = 1.3;
         double bc = 10;
         double be;
         double dm[8];
         double dc[8];
         double e[8];
         double step = 0.01;
         double best_error = 999999999;
         int best_error_i;
         int minimum_found = 0;

         double om[] = {0,1,1, 1, 0,-1,-1,-1};
         double oc[] = {1,1,0,-1,-1,-1, 0, 1};

       struct timespec start, finish;
           long long int time_elapsed;
           clock_gettime(CLOCK_MONOTONIC, &start);
           printf("\n===============================================================================\n");
         printf("!! OUTPUT FOR LINEAR REGRESSION !! \n");
         printf("===============================================================================\n\n");
       cudaError_t error;


       double *d_dm;
       double *d_dc;
       double *d_error_sum_arr;
       point_t *d_data;

       be= rms_error(bm,bc);

       error=cudaMalloc(&d_dm,(sizeof(double) * 8));
       if(error){
       fprintf(stderr,"cudaMalloc on d_dm returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }

       error=cudaMalloc(&d_dc,(sizeof(double) * 8));
       if(error){
       fprintf(stderr,"cudaMalloc on d_dc returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }

       error=cudaMalloc(&d_error_sum_arr,(sizeof(double) * 1000));
       if(error){
       fprintf(stderr,"cudaMalloc on d_error_sum_arr returned %d %s\n",error, //371
       cudaGetErrorString(error));
       exit(1);
       }

       error=cudaMalloc(&d_data,sizeof(data)); //376
       if(error){
       fprintf(stderr,"cudaMalloc on d_data returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }

       while(!minimum_found) {
           for(i=0;i<8;i++) {
       dm[i] = bm + (om[i] * step);
       dc[i]= bc + (oc[i] * step);
       }

        error = cudaMemcpy(d_dm,dm,(sizeof(double)*8), cudaMemcpyHostToDevice);
       if(error){
       fprintf(stderr,"cudaMemcpy to d_dm returned %d %s\n",error,
       cudaGetErrorString(error));
       }

        error = cudaMemcpy(d_dc,dc,(sizeof(double)*8), cudaMemcpyHostToDevice);
       if(error){
       fprintf(stderr,"cudaMemcpy to d_dc returned %d %s\n",error,
       cudaGetErrorString(error));
       }

       error = cudaMemcpy(d_data, data,sizeof(data), cudaMemcpyHostToDevice); //401
       if(error){
       fprintf(stderr,"cudaMemcpy to d_data returned %d %s\n",error,
       cudaGetErrorString(error));
       }

       for(i=0;i<8;i++){
       double h_error_sum_arr[1000];

       double error_sum_total;
       double error_sum_mean;

       d_rms_error <<<100,10>>>(&d_dm[i],&d_dc[i],d_error_sum_arr,d_data);
       cudaThreadSynchronize();
       error =cudaMemcpy(&h_error_sum_arr,d_error_sum_arr,(sizeof(double) *1000),
       cudaMemcpyDeviceToHost);
       if(error){
       fprintf(stderr,"cudaMemcpy to error_sum returned %d %s\n",error,
       cudaGetErrorString(error));
       }
       for(int j=0;j<n_data;j++){
       error_sum_total+= h_error_sum_arr[j];
       }
       error_sum_mean = error_sum_total / n_data;
       e[i] =sqrt(error_sum_mean);

       if(e[i] < best_error){
       best_error = e[i];
       error_sum_total +=h_error_sum_arr[i];
       }
       error_sum_mean = error_sum_total /n_data;//431
       e[i] =  sqrt(error_sum_mean); //432

       if(e[i]<best_error){ //434
       best_error = e[i];
       best_error_i = i;
       }
        error_sum_total = 0;  //438
       }
       if(best_error <be){
       be=best_error;
       bm =dm[best_error_i];
       bc= dc[best_error_i];
       }else {
       minimum_found = 1;
       }
       }


       error = cudaFree(d_dm);
       if(error){
       fprintf(stderr,"cudaFree on d_dm returned %d %s\n",error,
       cudaGetErrorString(error));  //453
       exit(1);
       }

       error = cudaFree(d_dc);
       if(error){
       fprintf(stderr,"cudaFree on d_dc returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }

       error = cudaFree(d_data);
       if(error){
       fprintf(stderr,"cudaFree on d_data returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }

       error = cudaFree(d_error_sum_arr);
       if(error){
       fprintf(stderr,"cudaFree on d_error_sum_arr returned %d %s\n",error,
       cudaGetErrorString(error));
       exit(1);
       }


       printf("minimum m,c is %lf,%lf with error %lf", bm, bc, be);

       clock_gettime(CLOCK_MONOTONIC, &finish);
         time_difference(&start, &finish, &time_elapsed);
         printf("\n\n===============================================================================\n");
         printf("!! TIME TAKEN FOR EXECUTION !! \n");
         printf("===============================================================================\n\n");
         printf("Nanoseconds: %lld\n", time_elapsed);
         printf("Seconds: %0.9lf\n", ((time_elapsed/1.0e9)));
         printf("Minutes: %0.4lf\n", ((time_elapsed/1.0e9)/60));
         printf("Hours: %0.2lf\n\n", ((time_elapsed/1.0e9)/3600));

       return 0;
       }

```

Insert a table that shows running times for the original and CUDA versions.

Attempt	Time taken for Execution (in seconds)	
	Original Program	CUDA Version
1	0.143629998	0.033120716
2	0.14021325	0.028139638
3	0.135532035	0.038288106
4	0.137495282	0.032101862
5	0.140809203	0.031927808
6	0.149402668	0.046372792
7	0.225641649	0.030621526
8	0.145377408	0.037252739
9	0.157061111	0.041136235
10	0.144941257	0.03598865
Mean Running Time	**0.152010386**	**0.035495007**

Write a short analysis of the results

Analysis:

The above table displays the difference in mean running time between the original version and the CUDA version of the linear regression program, where the CUDA version is 4.2 times faster than the original version of the program. The reason is simple, as there are 100 threads used in a single block, where there are 10 blocks i.e. thread is 100 times 10 in the GPU.