UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

6CS005 High Performance Computing

Assessment Portfolio

Learning Journal

Semester 1=2019-20

Student Id: 1929081

Student Name: Shanti Ghimire

Module Leader: Mr. Jnaneshwar Bohara

Group: C3G3

Cohort: 3

Submitted on: 05/01/2020

Table of Contents

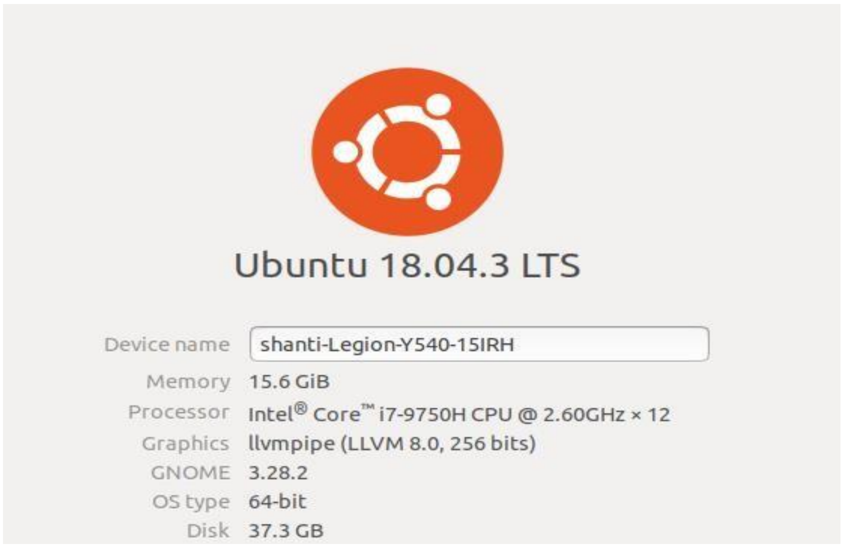
1

1. Introduction:

The programs are run on three different ways:

I. Posix
II. CUDA
III. MPI

Therefore, the programs are run on the desktop provided by the college. The device's configuration is mentioned below.



Ubuntu 18.04.3 LTS

Device name	shanti-Legion-Y540-15IRH
Memory	15.6 GiB
Processor	Intel® Core™ i7-9750H CPU @ 2.60GHz × 12
Graphics	llvmpipe (LLVM 8.0, 256 bits)
GNOME	3.28.2
OS type	64-bit
Disk	37.3 GB

Illustration 1: Computer Specification

Since the program needs to be ran multiple times to calculate the mean running time of the programs, two program which was taught in the tutorial class is used. The running time for all the program is calculated in seconds whereas the password cracking in Posix thread is calculated in nanoseconds as per the requirement. The file name and its uses mentioned below.

I. **mr.py:** used to run the password cracking program 10 times in one go.
II. **time_diff.c:** used to capture the execution time of the program.
III. **EncryptSHA512.c:** used to encrypt 3 initials and 2 digits password.

2. Posix:

2.1 Password Cracking:

a) The mean running time of the program after running 10 times.

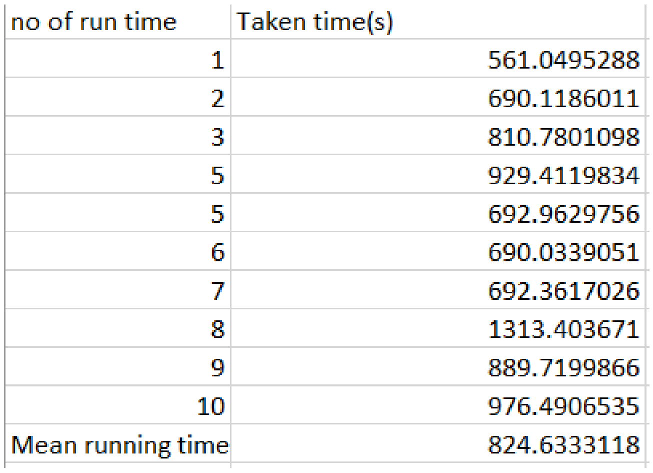
Answer:

Source code for an original program:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <crypt.h>
#include <time.h>

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


  Compile with:
    cc -o CrackAZ99-With-Data CrackAZ99-With-Data.c -lcrypt
Run with:

    ./CrackAZ99-With-Data > results.txt

****************************************************************************
/
int n_passwords = 4;

char *encrypted_passwords[] = {

"$6$KB$0G24VuNaA9ApVG4z8LkI/OOr9a54nBfzgQjbebhqBZxMHNg0HiYYf1Lx/HcGg6q1nnO
SArPtZYbGy7yc5V.wP/",

"$6$KB$WksuNcTfYjZWjDC4Zt3ZAmQ38OrsWwHyDgf/grFJ2Sgg.qpOz56lMpBVfWYdQZa9Pks
a2TJRVYVb3K.mbYx4Y1",

"$6$KB$UdJ/FGlqWHrXeWFVdjwqMel4WRTW93ai6K891ug/Td3NnAWj1AMMfZkQGut4Ia7hpWb
4ECic6xlvF.BGJdOj90",

"$6$KB$mV33QckPvVM55rLtO3QTXr5ib3rvmyndjSWLt0DZSOimZ0bM/djcZRyTY0fm25xKc/u
5b.aTNjV8mBxv9ESTL0",
};

/**
 Required by lack of standard function in C.
*/

void substr(char *dest, char *src, int start, int length){
  memcpy(dest, src + start, length);
  *(dest + length) = '\0';
}
void crack(char *salt_and_encrypted){
  int x, y, z;      // Loop counters
  char salt[7];     // String used in hashing the password. Need space for
\0
  char plain[7];    // The combination of letters currently being checked
  char *enc;        // Pointer to the encrypted password
  int count = 0;    // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='A'; x<='Z'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
        sprintf(plain, "%c%c%02d", x, y, z);
        enc = (char *) crypt(plain, salt);
        count++;
        if(strcmp(salt_and_encrypted, enc) == 0){
          printf("#%-8d%s %s\n", count, plain, enc);
        } else {
          printf(" %-8d%s %s\n", count, plain, enc);
        }
      }
    }
  }
  printf("%d solutions explored\n", count);
}
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[]){
  int i;
  struct timespec start, finish;
  long long int time_elapsed;
  clock_gettime(CLOCK_MONOTONIC, &start);
  for(i=0;i<n_passwords;i<i++) {
    crack(encrypted_passwords[i]);
  }
 clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));
  return 0;
}
```

Illustration 2: Password Cracking original program code

Mean Running time:



no of run time	Taken time(s)
1	561.0495288
2	690.1186011
3	810.7801098
5	929.4119834
5	692.9629756
6	690.0339051
7	692.3617026
8	1313.403671
9	889.7199866
10	976.4906535
Mean running time	824.6333118

Illustration 3: Password Cracking original program

b) Time estimation of how long the program would take to run on the same computer if the number of initials were increased to 3.

Answer:

The original program has two initials and 2 digits i.e. JB12. Its mean running time is 824.6333118s. But when the initials are increased by one. we have, JSB99, a 5 digits password and the added loop will run for 26 times since 26 alphabets. The running time of the 3 initials and 2 digits password would increase by 26 which help to compared to the 2 initials and 2 digits password. Therefore, the mean running time would be near to 26* 824.6333118s i.e. 21,440.466s.

c) Modify the program to crack the three-initials-two-digits password given in the Three_initials variable.

Answer:

Source code for a 3 initial password cracker:

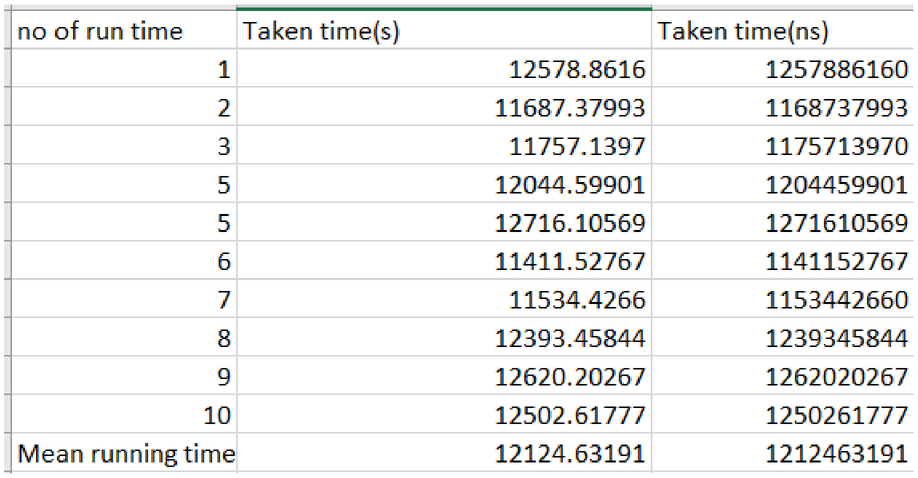
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <crypt.h>
#include <time.h>

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

  Compile with:
    cc -o CrackAZ99-With-Data 3initialpasswordcrack.c -lcrypt -pthread

    ./CrackAZ99-With-Data >crack.txt

************************************************************************
******/
int n_passwords = 4;

char *encrypted_passwords[] = {


"$6$KB$USbZToxSR.NNRAtXir98EwcU5ilQT0MNLvEIQiQeX7WpJEK/ey3U/L9DB1J5y.30LDI
ce1xu0RiS5WwY4vgI4/",

"$6$KB$9b40a.jXbR.UCKxKbcJCA8pDac43HeeNKh11haB1jdC9XVgC5RgeGrHQXC6Lug4TAHl
y4R3hiSlQx0AIhX51H1",

"$6$KB$TXpHr7EJl1nt0zn.PfixteNX02v9joyslUtz/OERahB/tS8WQlu7s43Kn75ou40JIgY
oOvVUD8GU4Fmevfpgm1",
"$6$KB$EQNNwMIddTvap2NhJEhL9p3DhryTPGc/onLK44dWJrpW/Cxh0Udu3dMk.vO79jQlsZa
4IYMNR/EVVDAKJVJje/",
};

/**
 Required by lack of standard function in C.
*/

void substr(char *dest, char *src, int start, int length){
  memcpy(dest, src + start, length);
  *(dest + length) = '\0';
}
void crack(char *salt_and_encrypted){
  int w, x, y, z;     // Loop counters
  char salt[7];    // String used in hashing the password. Need space

  char plain[7];   // The combination of letters currently being checked
  char *enc;       // Pointer to the encrypted password
  int count = 0;   // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);
        for(w='A'; w<='Z'; w++){
          for(x='A'; x<='Z'; x++){
            for(y='A'; y<='Z'; y++){
              for(z=0; z<=99; z++){
               sprintf(plain, "%c%c%c%02d", w, x, y, z);
               enc = (char *) crypt(plain, salt);
               count++;
               if(strcmp(salt_and_encrypted, enc) == 0){
                // printf("#%-8d%s %s\n", count, plain, enc);
               } else {
                 //printf(" %-8d%s %s\n", count, plain, enc);
               }
              }
            }
          }
        }
  printf("%d solutions explored\n", count);
}
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[]){

  struct timespec start, finish;
  long long int difference;
  int account = 0;
  clock_gettime(CLOCK_MONOTONIC, &start);

  int i;

  for(i=0;i<n_passwords;i<i++) {
    crack(encrypted_passwords[i]);
  }

  clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &difference);

  printf("Elapsed Time: %9.5lfs\n", difference/1000000000.0);
  printf("\n");
  return 0;
}
```

Illustration 4: 3initial password cracking

d) Write a short paragraph to compare the running time of your three_initials program with your earlier estimate. If your estimate was wrong explained why you think that is.

Answer:

Mean Run time:



no of run time	Taken time(s)	Taken time(ns)
1	12578.8616	1257886160
2	11687.37993	1168737993
3	11757.1397	1175713970
5	12044.59901	1204459901
5	12716.10569	1271610569
6	11411.52767	1141152767
7	11534.4266	1153442660
8	12393.45844	1239345844
9	12620.20267	1262020267
10	12502.61777	1250261777
Mean running time	12124.63191	1212463191

Illustration 5: 3initial password cracking mean time

I had estimated that the program with 3 initials and 2 digit would have running time increased by 26 times i.e 21,440.466s. Since the running time for the program was nearly increased by 26 times, therefore, my estimation was nearly the same i.e. 12124.63191s.

e) Modify the original version of the program to run on 2 threads.

Answer:

Source code for a Multithread password cracker:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <crypt.h>
#include <time.h>
#include <pthread.h>

/***********************************************************************
  Compile with:
    cc -o CrackAZ99-Posix CrackAZ99-Posix-With-Thread.c -lcrypt -pthread
  Run with:
    ./CrackAZ99-Posix > results.txt

************************************************************************/
int count = 0; //count the solution explored

int n_passwords = 4;

char *encrypted_passwords[] = {


"$6$KB$0G24VuNaA9ApVG4z8LkI/OOr9a54nBfzgQjbebhqBZxMHNg0HiYYf1Lx/HcGg6q1nnO
SArPtZYbGy7yc5V.wP/",

"$6$KB$WksuNcTfYjZWjDC4Zt3ZAmQ38OrsWwHyDgf/grFJ2Sgg.qpOz56lMpBVfWYdQZa9Pks
a2TJRVYVb3K.mbYx4Y1",

"$6$KB$UdJ/FGlqWHrXeWFVdjwqMel4WRTW93ai6K891ug/Td3NnAWj1AMMfZkQGut4Ia7hpWb
4ECic6xlvF.BGJdOj90",

"$6$KB$mV33QckPvVM55rLtO3QTXr5ib3rvmyndjSWLt0DZSOimZ0bM/djcZRyTY0fm25xKc/u
5b.aTNjV8mBxv9ESTL0"
```

```
      };

/**
 Required by lack of standard function in C.
*/

void substr(char *dest, char *src, int start, int length){
  memcpy(dest, src + start, length);
  *(dest + length) = '\0';
}
void *crack1(void *salt_and_encrypted){
  int x, y, z;      // Loop counters
  char salt[7];     // String used in hashing the password. Need space for
\0
  char plain[7];    // The combination of letters currently being checked
  char *enc;        // Pointer to the encrypted password

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='A'; x<='M'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
        sprintf(plain, "%c%c%02d", x, y, z);
        enc = (char *) crypt(plain, salt);
        count++;
        if(strcmp(salt_and_encrypted, enc) == 0){
          printf("#%-8d%s %s\n", count, plain, enc);
        } else {
          printf(" %-8d%s %s\n", count, plain, enc);
        }
      }
    }

  }
  pthread_exit(NULL);
}
```

```
void *crack2(void *salt_and_encrypted){
  int x, y, z;     // Loop counters
  char salt[7];    // String used in hashing the password. Need space for
\0
  char plain[7];   // The combination of letters currently being checked
  char *enc;       // Pointer to the encrypted password

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='N'; x<='Z'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
        sprintf(plain, "%c%c%02d", x, y, z);
        enc = (char *) crypt(plain, salt);
        count++;

        if(strcmp(salt_and_encrypted, enc) == 0){
          printf("#%-8d%s %s\n", count, plain, enc);
        } else {
          printf(" %-8d%s %s\n", count, plain, enc);
        }
      }
    }
  }

pthread_exit(NULL);
}

//function to calculate running time of crack function
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[]){
  int i;
  struct timespec start, finish;
  long long int time_elapsed;

  pthread_t thread_1, thread_2;
  int t1, t2;

  clock_gettime(CLOCK_MONOTONIC, &start);

  for(int i =0; i<n_passwords; i++){
  t1 = pthread_create(&thread_1, NULL, crack1,(void
*)encrypted_passwords[i]);
  if(t1){
  printf("Thread creation failed: %d\n", t1);
  }
 }
  for(int i =0; i<n_passwords; i++){
  t2 = pthread_create(&thread_2, NULL, crack2,(void
*)encrypted_passwords[i]);
  if(t2){
  printf("Thread creation failed: %d\n", t2);
  }
 }
  pthread_join(thread_1, NULL);
  pthread_join(thread_2, NULL);

  clock_gettime(CLOCK_MONOTONIC, &finish);
  printf("%d solutions explored\n", count);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns\n", time_elapsed);
 pthread_exit(NULL);
}
```

Illustration 6: Multithread password cracking

Analysis the code:

The modified program file is available in Folder POSIX Threads -> Password cracking with file named as ***CrackAZ99-Posix-With-Thread.c.***

A simple modification is made to the original program.

I. Two pthread is created is created in a loop until 4 passwords are cracked:

for(i=0;i<n_passwords; i++) {

a) Pthread 1: ***pthread_create(&t1, NULL, kernel_function_1, encrypted_passwords[i]);***

b) Pthread 2: ***pthread_create(&t1, NULL, kernel_function_2, encrypted_passwords[i]);***

ii. Two pthread function is created;

a) Pthread Function 1: ***void *kernel_function_1(void *salt_and_encrypted){***

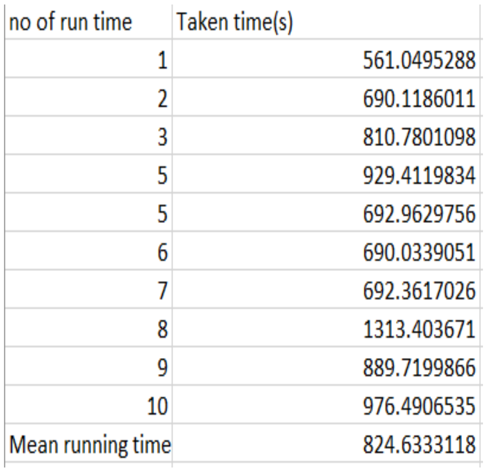
   It runs the loop from A-M : ***for(x='A'; x<='M'; x++){***

b) Pthread Function 2: ***void *kernel_function_2(void *salt_and_encrypted){***

   It runs the loop from N-Z ***: for(x='A'; x<='M'; x++){***

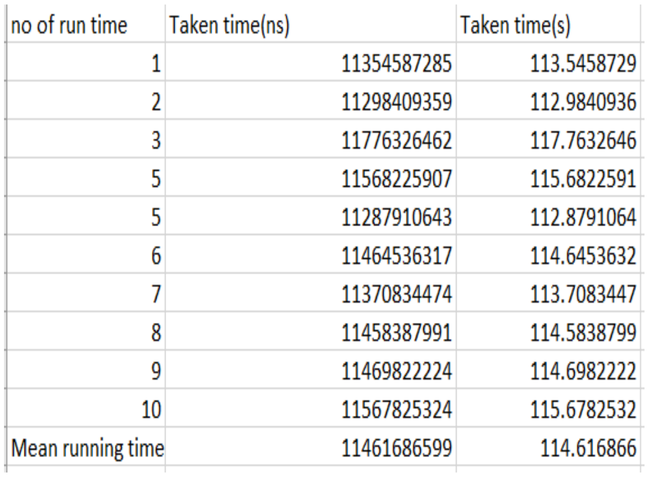
f) Comparison of the mean running time of the original program with the mean running time of the multithread version.

Answer:



no of run time	Taken time(s)
1	561.0495288
2	690.1186011
3	810.7801098
5	929.4119834
5	692.9629756
6	690.0339051
7	692.3617026
8	1313.403671
9	889.7199866
10	976.4906535
Mean running time	824.6333118

Illustration 7: Password Cracking original



no of run time	Taken time(ns)	Taken time(s)
1	11354587285	113.5458729
2	11298409359	112.9840936
3	11776326462	117.7632646
5	11568225907	115.6822591
5	11287910643	112.8791064
6	11464536317	114.6453632
7	11370834474	113.7083447
8	11458387991	114.5838799
9	11469822224	114.6982222
10	11567825324	115.6782532
Mean running time	11461686599	114.616866

Illustration 8: Password Cracking multithread

The mean running time of the original program is comparatively more than the mean running time of the multithread version by 50%. The original program cracks the password one at a time. But the program with multithread version with two threads: kernel_function_1 and kernel_function_2, cracks the password in parallel which takes less time than the original program.

2.1.1 Posix Password cracking Code Analysis:

mr.py: used to run the password cracking program 10 times in one go.
time_diff.c: used to capture the execution time of the program.
EncryptSHA512.c: used to encrypt 3 initials and 2 digits password.
Char Salt[7]: character array used in hashing password and salt will be the result of crypt function.
Char plain[7]:combination of letter for checking
Char* enc: pointer to the encrypted password

Clock getting(CLOCK_MONOTONIC, &start): getting execution time of program with timw difference both nanosecond and second

Pthread_create(): the Pthread_create() function starts a new thread in the calling process.

Pthread_join(): this function work on the waits for the thread specified by the thread to terminate.

Crypt(): This function encrypts the string pointed to by key using first two charcter of the sting pointed to by salt to perturb the encryption algorithm.

Encrypt(): This functions encrypts or decrypts depending on the value of the edflag arguments the string pointed to by block using the encryption key set by the setkey() function.

2.2 Image Processing:

a) The resulting image of the assigned program.

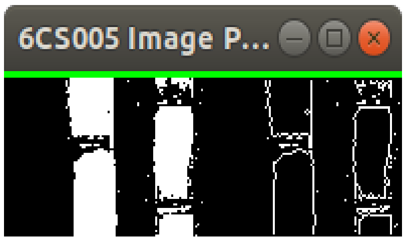


Illustration 9:Ouput of edge detection (Original Program)

Source code for the above image or original program:

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

/***************************************************************************
  To compile with:
    cc -o ip_coursework ip_coursework_034.c -lglut -lGL -lm
 Run with:
  ./ip_coursework


***************************************************************************/
#define width 100
#define height 72

unsigned char image[], results[width * height];

void detect_edges(unsigned char *in, unsigned char *out) {
  int i;
  int n_pixels = width * height;

  for(i=0;i<n_pixels;i++) {
    int x, y; // the pixel of interest
    int b, d, f, h; // the pixels adjacent to x,y used for the calculation
    int r; // the result of calculate

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

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

      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);
  struct timespec start, finish;
  long long int time_elapsed;

  clock_gettime(CLOCK_MONOTONIC, &start);
  detect_edges(image, results);


  clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));



  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;
}

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

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  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,
  255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,
  0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,255,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,0,255,255,0,0,255,255,255,
  255,255,255,0,255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,
  255,255,255,255,0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  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,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  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,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,0,255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,255,
  0,0,255,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,0,
  0,0,0,255,255,0,0,0,255,0,0,0,0,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,0,0,0,0,0,255,255,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,255,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,0,0,0,0,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,255,255,255,0,0,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,0,0,0,255,255,255,0,255,255,0,
  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,
  0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,0,0,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0
};
```

Illustration 10: Image Processing Original Program

b) Modified program for edge detector to process 4 pixels in parallel using striding technique.

Answer:

The modified program file is available in Folder POSIX Threads -> Image processing with file named as EDgeDetectionIMgeWithmultiThread.c.

A simple modification is made to the original program. The program runs in thread. simple modification is made to the original program.

i. Structure is created for striding technique:
Typedef struct arguments_t {int start; int stride;} arguments_t;

ii. Pthread function is created: **void *detector (void *args){**

iii. 4 Pthread is created: ***pthread_t t1, t2, t3, t4;***

iv. All the thread executes the same function:

- ➢ ***pthread_create(&t1, NULL, (void*)detector, &t1_arguments);***
- ➢ ***pthread_create(&t2, NULL, (void*)detector, &t2_arguments);***
- ➢ ***pthread_create(&t3, NULL, (void*)detector, &t3_arguments);***
- ➢ ***pthread_create(&t4, NULL, (void*)detector, &t4_arguments);***

Source code for your multithreaded edge detector:

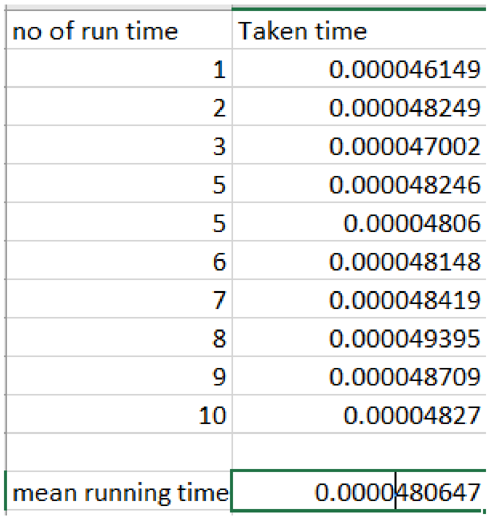
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <malloc.h>
#include <signal.h>
#include <ctype.h>
#include <errno.h>
#include <sys/stat.h>
#include <string.h>
#include <time.h>
#include <pthread.h>
#include <math.h>
#include "time_diff.h"
/**************************************************************************
  To compilewith:
    cc -o EDgeDetectionIMgeWithmultiThread
EDgeDetectionIMgeWithmultiThread.c time_diff.c -lglut -lGL -lm -pthread
 Run with:
 ./ EDgeDetectionIMgeWithmultiThread

**************************************************************************
****/
#define width 100
#define height 72
typedef struct arguments_t {
  int start;
  int stride;
} arguments_t;
unsigned char image[], results[width * height];
```

```
void detect_edges(unsigned char *in, unsigned char *out, arguments_t
*args) {
  int i;
  int n_pixels = width * height;

  for(i=args->start;i<n_pixels;i+=args->stride){
    int x, y; // the pixel of interest
    int b, d, f, h; // the pixels adjacent to x,y used for the calculation
    int r; // the result of calculate

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

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

      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 *detector(void *args){
       detect_edges(image,results,args);
}

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 main(int argc, char **argv) {

  struct timespec start, finish;
  long long int time_elapsed;

  clock_gettime(CLOCK_MONOTONIC, &start);
  signal(SIGINT, sigint_callback);

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

  pthread_t t1, t2, t3, t4;

  arguments_t t1_arguments;
  t1_arguments.start = 0;
  t1_arguments.stride = 4;

  arguments_t t2_arguments;
  t2_arguments.start = 1;
  t2_arguments.stride = 4;

  arguments_t t3_arguments;
  t3_arguments.start = 2;
  t3_arguments.stride = 4;

  arguments_t t4_arguments;
  t4_arguments.start = 3;
  t4_arguments.stride = 4;

  pthread_create(&t1, NULL, detector, &t1_arguments);
  pthread_create(&t2, NULL, detector, &t2_arguments);
  pthread_create(&t3, NULL, detector, &t3_arguments);
  pthread_create(&t4, NULL, detector, &t4_arguments);

  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  pthread_join(t3, NULL);
  pthread_join(t4, NULL);
```

```
 clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));
  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;

}

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

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  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,
  255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,
  0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,255,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,0,255,255,0,0,255,255,255,
  255,255,255,0,255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,
  255,255,255,255,0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  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,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  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,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,0,255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,255,
  0,0,255,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,0,
  0,0,0,255,255,0,0,0,255,0,0,0,0,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,0,0,0,0,0,255,255,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,255,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,0,0,0,0,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,255,255,255,0,0,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,0,0,0,255,255,255,0,255,255,0,
  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,
  0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,0,0,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0
};
```
Illustration 11: Image Processing Posix Thread

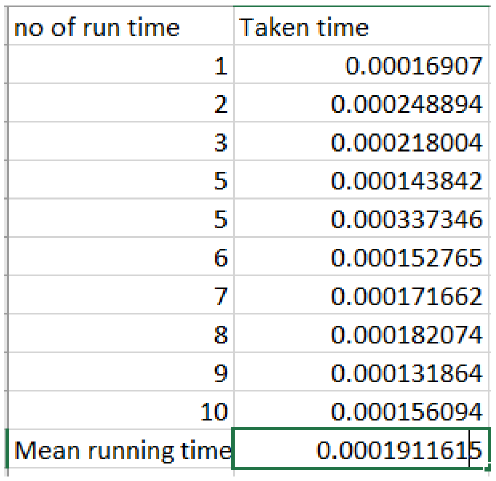
c) Compare the relative running times of the original edge detection program, with the multithread one.

Answer:



no of run time	Taken time
1	0.000046149
2	0.000048249
3	0.000047002
5	0.000048246
5	0.00004806
6	0.000048148
7	0.000048419
8	0.000049395
9	0.000048709
10	0.00004827
mean running time	0.0000480647

Illustration 12: Original time



no of run time	Taken time
1	0.00016907
2	0.000248894
3	0.000218004
5	0.000143842
5	0.000337346
6	0.000152765
7	0.000171662
8	0.000182074
9	0.000131864
10	0.000156094
Mean running time	0.0001911615

Illustration 13: Image with *Posix Thread*

The original program runs faster in comparison to the program with striding technique. As the striding processes 4 pixels in parallel which takes extra time to combine the pixels and put it together. Therefore, it takes extra time than the original program.

2.2.1 Posix Image Processing Code Analysis:

The main aim of this code is to know how to computer stores and manipulates two-dimensional array of cells called image. Picture made up of pixels which made up of horizontal and vertical coordinated x and y. Two time_spec start and finish along with the time elapsed is declared in declaration phase to access the start time, finish time and the time difference respectively. Clock_get time function is used to get the time from the system which takes two parameters CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time.

Structure is created for striding technique:

Typedef struct arguments_t {int start; int stride;} arguments_t;

Pthread function is created: **void *detector (void *args){**

4 Pthread is created: ***pthread_t t1, t2, t3, t4;***

All the thread executes the same function:

- ***pthread_create(&t1, NULL, (void*)detector, &t1_arguments);***
- ***pthread_create(&t2, NULL, (void*)detector, &t2_arguments);***
- ***pthread_create(&t3, NULL, (void*)detector, &t3_arguments);***
- ***pthread_create(&t4, NULL, (void*)detector, &t4_arguments);***

2.3 Linear regression:

a) Do a scatter plot of your dataset and put it in your portfolio.

Answer:

```
plt.scatter(dataset.x, dataset.y)
plt.show()
```

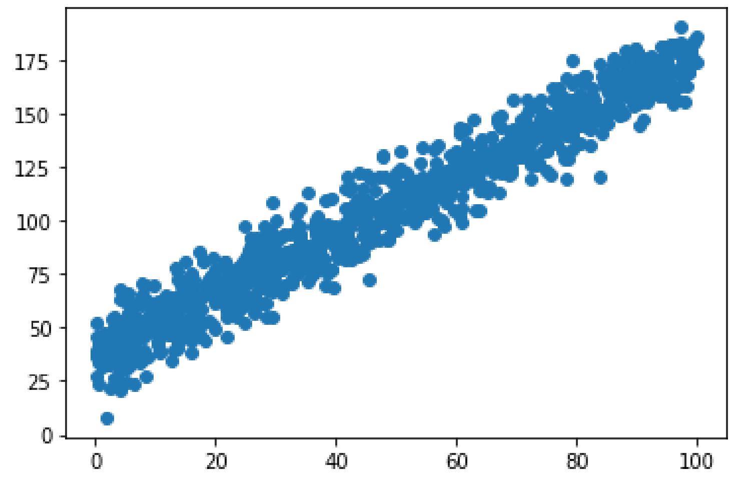


Illustration 14: Data plot of the given data set

b) Program and its plotting results with three guesses at the optimum values for m and c.

In the graph below,

- ➢ Red color represents first guess with m=0, c=35
- ➢ Yellow color represents second guess with m=0, c=36
- ➢ Blue color represents third guess with m=0, c=38

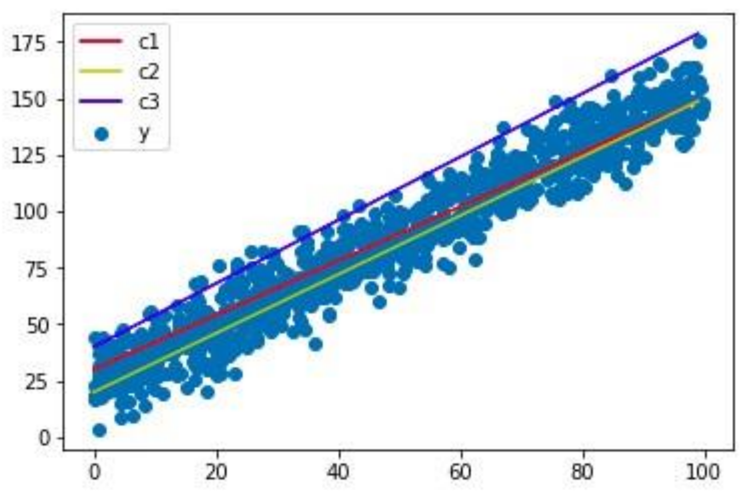


Illustration 15: Data plot of three guesses of m and c

c) Run the program to see what solution it finds. Overlay the line that was found by the program on to a dataset scatter plot and comment on the solution.

Answer:

```
#include <stdio.h>
#include <math.h>

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

 * To compile:
 *   cc -o 34_b 34_b.c -lm
 *
 * To run:
 *   ./lr_coursework
 *****************************************************************************/


int main(int argc, char **argv) {
  int i;
  double m;
  double c;
  double x;
  double y;

  if(argc != 3) {
    fprintf(stderr, "You need to specify a slope and intercept\n");
    return 1;
  }

  sscanf(argv[1], "%lf", &m);
  sscanf(argv[2], "%lf", &c);
  printf("x,y\n");
  for(i=0; i<100; i++) {
    x = i;
    y = (m * x) + c;
    printf("%0.2lf,%0.2lf\n", x, y);
  }

  return 0;
}
```

Illustration 16: Data plot of three guesses of m and c program

Result:

- ➢ Red color represents first guess with m=0, c=35
- ➢ Yellow color represents second guess with m=0, c=36
- ➢ Blue color represents third guess with m=0, c=38
- ➢ Cyan color represents best m=1.41, c=35.75

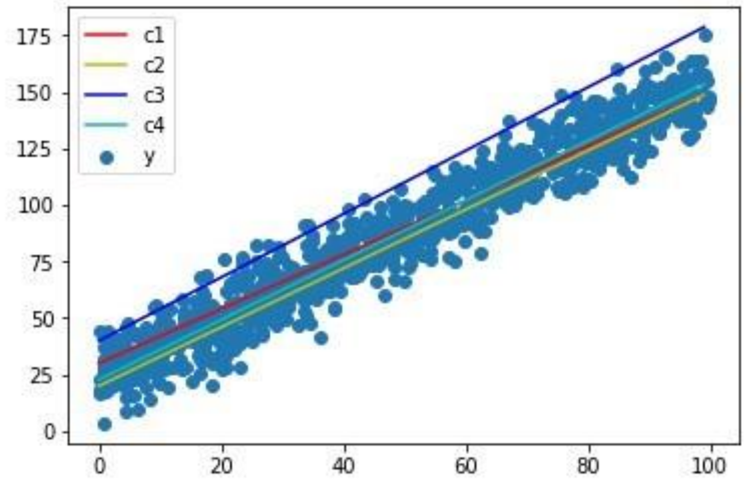


Illustration 17: Data plot of three guesses of m and c with best m and c

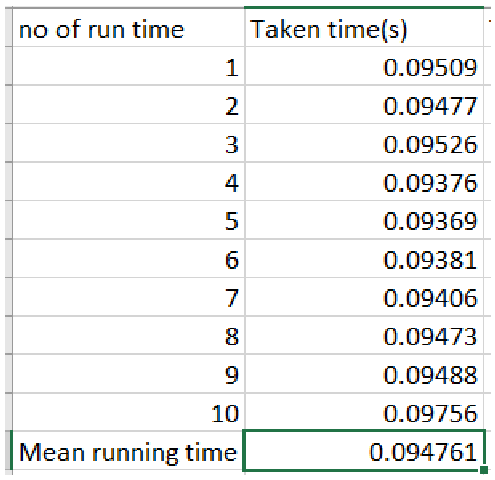
d) Remove any extraneous print statements from the program and find its mean running time.

Answer:

Source code for an original program:

```
#include <stdio.h>
#include <math.h>

/**********************************************************************
 *
 * To compile:
 *   cc -o lr_coursework 34.c -lm
 *
 * To run:
 *   ./lr_coursework
 *
**********************************************************************
***/

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

int n_data = 1000;
point_t data[];

double 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);
}

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};

  be = rms_error(bm, bc);

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

    for(i=0;i<8;i++) {
      e[i] = rms_error(dm[i], dc[i]);
      if(e[i] < best_error) {
        best_error = e[i];
        best_error_i = i;
      }
    }

    printf("best m,c is %lf,%lf with error %lf in direction %d\n",
      dm[best_error_i], dc[best_error_i], best_error, best_error_i);
    if(best_error < be) {
      be = best_error;
      bm = dm[best_error_i];
      bc = dc[best_error_i];
    } else {
      minimum_found = 1;
    }
  }
  printf("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);

  return 0;
}

point_t data[] = {
 {65.11,126.40},{76.79,149.00},{76.93,162.00},{65.24,113.46},
  {78.54,145.93},{84.60,161.77},{85.60,162.58},{82.32,152.21},
  {78.17,144.80},{69.47,142.78},{82.72,156.05},{11.56,52.20},
  {66.15,122.01},{75.13,145.75},{ 8.11,36.01},{71.58,150.44},
  {23.30,70.06},{42.59,86.42},{39.11,76.86},{ 8.77,36.29},
  {83.41,152.32},{ 3.44,36.86},{72.15,126.11},{66.29,129.15},
  {28.93,92.24},{91.62,172.01},{ 0.39,40.02},{55.24,104.88},
  {44.96,90.98},{89.66,170.29},{29.39,86.66},{56.19,109.96},
  {79.43,153.15},{54.27,110.73},{ 9.28,54.91},{31.16,76.74},
  {20.00,49.29},{67.25,122.68},{95.64,182.45},{66.03,128.30},
  {36.60,94.60},{83.93,120.16},{22.67,76.02},{81.17,164.59},
  {84.70,147.15},{34.58,87.81},{ 0.26,39.25},{82.07,149.43},
  { 2.63,41.39},{ 1.74, 7.95},{70.98,133.50},{16.65,49.48},
  {27.85,61.85},{55.84,105.63},{81.77,153.60},{19.81,61.26},
  {28.19,97.35},{ 2.62,32.52},{60.42,123.43},{53.67,118.83},
  {92.67,163.43},{ 4.09,30.06},{31.35,78.55},{54.79,103.97},
  {89.15,163.38},{20.35,66.02},{28.55,88.62},{11.66,58.29},
  {89.90,154.40},{ 0.14,51.92},{ 4.75,37.69},{53.83,108.99},
  {62.17,127.68},{79.10,133.64},{24.19,68.78},{51.41,100.86},
  {44.52,92.93},{23.02,66.51},{98.60,181.12},{ 6.05,48.82},
  {62.79,147.70},{ 5.06,50.58},{85.40,155.28},{12.33,60.17},
  {49.62,118.33},{ 9.03,48.29},{45.21,88.73},{28.22,55.37},
  {91.32,165.67},{ 6.74,44.19},{46.03,93.83},{69.75,139.69},
  { 2.15,40.31},{95.82,160.20},{ 6.64,54.91},{75.25,148.74},
  {39.64,68.97},{ 5.55,66.26},{90.53,155.37},{39.95,91.42},
  {68.89,132.98},{33.52,78.37},{15.84,38.51},{72.73,139.50},
  {21.54,73.78},{ 4.64,47.34},{66.57,132.87},{27.38,71.86},
  {93.83,181.33},{75.83,161.75},{26.47,56.70},{84.23,151.43},
  { 0.43,43.29},{88.50,160.27},{66.15,129.59},{78.31,141.68},
  {36.90,101.61},{71.78,139.52},{90.37,173.79},{ 0.58,45.98},
  {67.63,131.85},{57.43,100.37},{88.43,161.15},{74.83,132.98},
  {29.31,54.66},{79.06,146.78},{54.41,120.80},{51.76,108.96},
  {11.80,65.51},{38.19,90.48},{18.40,71.77},{76.29,148.07},
  {75.30,135.15},{59.56,126.34},{32.71,86.25},{42.35,116.15},
  { 4.85,38.50},{ 3.14,50.60},{48.27,90.59},{34.96,88.02},
  {10.03,50.01},{ 5.51,40.83},{68.32,136.16},{74.87,134.02},
  { 1.56,47.50},{19.52,72.68},{ 9.10,52.12},{50.79,102.10},
  {53.11,105.38},{94.93,174.68},{16.03,44.26},{13.26,49.58},
  { 3.24,46.86},{77.38,158.65},{21.57,62.81},{41.63,89.86},
  {13.55,59.72},{25.43,71.35},{86.73,166.79},{77.15,149.52},
  {26.47,64.94},{48.65,92.62},{33.66,75.10},{25.20,63.45},
  {25.46,86.18},{70.52,147.39},{98.32,175.47},{23.09,62.68},
  {48.90,118.74},{69.07,141.45},{50.54,132.25},{55.80,119.88},
  {25.65,92.01},{54.39,112.81},{79.86,165.28},{95.98,154.86},
  {48.14,108.06},{36.33,88.43},{ 6.34,35.96},{86.04,151.77},
  {57.03,116.32},{97.95,180.12},{29.66,73.83},{12.52,35.04},
  {43.93,83.56},{33.63,78.69},{64.00,128.13},{14.49,50.14},
  {49.66,112.89},{82.54,162.20},{81.92,143.19},{28.07,78.90},
  {14.26,47.92},{23.97,63.31},{27.69,73.01},{78.13,119.54},
  {34.43,82.48},{66.13,123.89},{61.84,135.81},{17.03,57.30},
  { 5.61,52.51},{34.44,88.49},{17.81,81.52},{34.26,79.71},
  {93.17,161.29},{ 8.10,39.44},{93.51,158.23},{61.48,133.51},
  {27.22,71.93},{17.11,50.22},{27.73,81.68},{16.07,61.27},
  {63.81,122.63},{ 0.27,36.83},{62.21,120.74},{42.36,85.01},
  {60.61,143.23},{68.59,121.10},{28.48,68.27},{23.39,71.50},
  {93.40,162.60},{50.72,114.87},{24.53,80.83},{92.00,160.38},
  {79.29,175.12},{28.84,78.42},{13.79,44.14},{23.18,62.24},
  {69.07,122.23},{41.93,120.63},{62.32,125.07},{72.39,136.08},
  {41.86,92.41},{ 2.35,21.43},{56.14,133.02},{33.91,90.07},
  {13.68,76.01},{14.55,71.51},{73.79,152.07},{33.47,97.28},
  {31.12,65.68},{ 4.33,41.19},{22.94,58.10},{85.12,160.37},
  {80.26,154.39},{37.01,78.54},{ 6.94,38.10},{ 7.83,60.51},
  {42.44,90.12},{26.69,91.63},{99.36,184.47},{ 9.33,55.05},
  {16.87,63.97},{32.41,80.49},{36.34,77.15},{59.91,122.62},
  { 7.35,34.97},{99.21,183.69},{34.07,97.06},{43.85,102.96},
  {55.20,111.99},{ 3.95,41.47},{26.71,82.57},{16.69,64.61},
  {38.32,96.85},{76.67,136.21},{86.22,175.61},{35.18,71.39},
  {57.39,117.85},{72.12,139.30},{90.19,173.32},{97.26,163.25},
  {82.08,135.04},{40.69,96.78},{25.49,75.76},{83.38,149.39},
  {63.64,135.54},{90.52,166.25},{79.96,154.68},{45.70,107.42},
  {15.54,61.07},{97.10,170.63},{41.10,87.33},{35.86,74.41},
  {57.22,120.65},{16.28,64.80},{46.33,97.53},{31.84,83.82},
  {90.15,177.85},{13.39,77.75},{ 8.25,26.83},{91.74,155.44},
  {11.65,61.09},{26.30,82.75},{61.72,128.34},{76.94,152.59},
  {26.70,81.25},{ 6.11,39.57},{97.46,172.22},{13.50,43.37},
  {16.46,54.81},{44.36,84.30},{45.83,105.17},{41.47,105.60},
  {31.72,82.66},{58.79,113.04},{95.35,168.80},{27.91,73.77},
  {61.28,126.49},{ 1.18,32.24},{ 4.17,41.67},{79.08,142.98},
  { 4.80,37.58},{94.98,160.79},{37.47,80.15},{ 6.82,53.58},
  {57.54,118.51},{73.31,139.05},{91.40,166.67},{98.07,162.54},
  { 3.87,41.53},{63.71,130.41},{75.78,137.34},{56.32,122.22},
  { 8.03,63.52},{22.60,60.09},{94.56,158.12},{16.10,72.60},
  {82.22,155.28},{57.63,114.94},{55.26,118.38},{73.52,126.54},
  {59.13,123.52},{81.11,167.56},{68.73,123.16},{43.78,122.00},
  {27.48,70.82},{43.92,110.09},{ 7.04,44.70},{91.03,147.13},
  {44.55,114.71},{68.40,122.77},{ 6.31,55.15},{12.03,51.80},
  {26.62,77.09},{12.90,60.85},{41.47,115.00},{75.98,156.05},
  {62.84,118.68},{ 3.19,54.74},{74.93,132.72},{89.37,170.57},
  {57.12,114.25},{63.07,104.90},{60.20,129.41},{36.04,82.56},
  {66.43,133.16},{ 7.01,45.98},{87.68,162.25},{36.24,86.35},
  {60.38,140.39},{ 3.56,41.80},{65.74,138.56},{16.34,61.32},
  {34.00,105.97},{77.42,132.46},{61.79,135.32},{61.13,116.51},
  {90.22,159.21},{68.78,137.53},{48.35,110.41},{58.48,110.06},
  { 0.04,45.54},{87.14,152.92},{73.42,154.90},{48.73,105.35},
  {36.26,97.55},{50.34,107.46},{95.65,162.87},{11.76,50.19},
  {12.83,58.44},{ 7.59,65.35},{51.44,109.31},{15.39,65.30},
  {83.69,147.00},{75.86,139.91},{25.87,87.89},{ 0.79,40.68},
  { 4.87,62.14},{64.71,126.73},{60.94,111.46},{82.18,142.90},
  {21.67,55.08},{33.20,93.76},{11.93,64.07},{10.59,39.64},
  {20.90,63.85},{21.47,81.20},{15.02,80.95},{67.04,112.88},
  { 0.78,34.00},{24.49,77.44},{ 0.49,33.31},{70.88,138.01},
  {33.07,81.36},{74.11,139.95},{ 1.26,40.24},{26.68,81.03},
  {81.37,155.39},{89.28,163.35},{82.92,150.44},{88.94,166.64},
  {14.35,63.72},{58.92,119.69},{47.88,100.06},{73.51,145.81},
  {21.11,79.13},{33.98,87.52},{87.88,158.32},{24.47,79.36},
  {34.00,92.04},{42.25,105.99},{20.75,64.71},{65.45,131.84},
  {10.51,38.23},{36.26,87.53},{70.72,129.61},{62.87,129.51},
  {88.93,170.73},{50.72,124.30},{52.28,104.85},{ 1.38,36.72},
  {93.31,171.99},{56.94,135.14},{99.80,174.15},{ 2.74,46.10},
  {42.39,81.82},{42.07,102.65},{ 3.06,34.94},{69.46,129.37},
  {43.33,104.41},{69.47,129.06},{68.70,134.00},{95.28,179.59},
  {66.66,132.99},{14.40,70.02},{39.22,77.48},{ 5.87,32.85},
  {71.70,144.19},{69.40,140.87},{51.33,113.42},{38.58,97.94},
  {70.18,129.30},{28.53,76.70},{11.77,52.53},{16.26,69.96},
  {86.39,161.92},{14.35,60.74},{17.32,53.69},{60.10,117.59},
  {55.86,117.63},{ 9.26,58.48},{48.28,100.74},{79.81,149.90},
  {59.38,105.17},{72.31,119.55},{41.23,91.88},{70.20,136.75},
  {82.58,158.00},{72.29,130.76},{10.80,60.44},{81.60,167.47},
  {57.21,120.91},{83.97,150.98},{78.97,152.79},{78.71,146.61},
  {98.28,172.82},{39.89,87.34},{92.46,169.18},{29.94,69.32},
  {64.78,127.52},{52.32,113.01},{29.54,78.81},{46.04,97.77},
  {97.71,155.81},{90.08,172.72},{59.58,117.24},{74.61,149.80},
  {64.20,116.45},{42.14,102.85},{44.27,107.95},{97.48,174.75},
  {84.52,164.71},{19.46,51.01},{87.05,166.73},{28.47,61.63},
  { 0.59,40.82},{49.83,100.11},{25.05,62.68},{20.32,62.69},
  {56.64,126.50},{15.41,59.79},{98.36,173.55},{57.73,97.55},
  {29.88,86.10},{26.44,65.10},{92.31,174.76},{10.74,49.39},
  {88.32,163.82},{71.67,156.47},{94.40,181.01},{16.04,55.25},
  {76.78,141.43},{81.76,146.31},{81.41,144.40},{ 7.59,65.42},
  {29.83,93.19},{71.85,128.96},{17.45,45.91},{66.78,148.29},
  {87.12,149.00},{55.37,115.72},{ 8.34,69.56},{10.30,58.08},
  {26.49,79.04},{33.65,83.88},{82.84,146.92},{74.19,132.28},
  {47.38,105.49},{95.30,172.52},{ 8.60,54.66},{38.14,69.96},
  {14.06,65.00},{27.26,74.14},{31.28,82.60},{24.83,97.76},
  {53.38,101.53},{26.88,87.40},{77.79,153.42},{ 4.60,53.07},
  {36.70,76.53},{33.96,93.76},{64.04,116.47},{73.85,156.81},
  {31.88,85.18},{10.44,62.68},{41.55,99.48},{31.20,94.01},
  {69.63,137.46},{30.90,92.46},{54.24,103.71},{82.12,149.86},
  {57.43,119.86},{16.83,61.60},{38.45,74.99},{ 9.38,60.12},
  {18.91,53.31},{65.28,115.98},{52.45,119.66},{ 4.88,52.30},
  {49.92,95.81},{60.44,126.35},{25.07,83.25},{58.21,108.57},
  {28.81,94.01},{44.94,99.55},{35.75,79.36},{95.26,164.51},
  {35.14,113.38},{ 6.95,51.77},{37.56,81.49},{27.79,72.55},
  {68.04,129.74},{19.46,82.90},{13.49,40.02},{40.81,92.01},
  {44.13,86.46},{90.16,178.62},{82.34,153.55},{92.32,165.64},
  {78.20,166.78},{24.76,66.00},{91.00,175.56},{85.94,172.81},
  {98.74,178.91},{ 7.47,34.02},{37.28,85.95},{ 8.94,54.67},
  {31.78,85.85},{31.71,82.87},{44.29,96.83},{21.85,68.96},
  {15.20,48.69},{ 9.51,58.55},{14.53,53.46},{87.25,166.09},
  {35.25,79.77},{45.43,106.79},{16.24,69.34},{61.36,142.83},
  {99.33,176.18},{ 2.66,42.92},{42.69,105.85},{69.04,124.32},
  {62.77,125.38},{87.76,149.94},{68.38,124.70},{44.95,109.62},
  { 8.36,63.51},{29.47,75.02},{42.49,87.92},{29.05,95.14},
  { 1.36,43.70},{60.36,102.16},{23.57,54.88},{30.84,80.74},
  {10.19,42.03},{97.59,177.44},{36.08,89.31},{21.74,45.86},
  {58.56,113.61},{34.10,92.54},{87.76,174.79},{43.42,107.45},
  {55.01,110.06},{45.87,119.35},{21.24,61.64},{ 0.63,23.13},
  {44.94,99.54},{ 5.22,47.01},{ 1.71,42.19},{92.32,159.09},
  {28.15,76.89},{77.98,128.92},{40.11,84.47},{80.44,144.10},
  {21.62,80.78},{27.18,70.12},{80.83,148.92},{65.54,132.52},
  {69.13,124.43},{26.54,59.95},{ 0.13,36.97},{24.07,70.64},
  {27.58,70.42},{45.07,121.14},{11.82,46.41},{81.39,156.60},
  {49.46,95.96},{56.25,93.87},{92.93,167.21},{85.35,169.34},
  {32.46,93.55},{37.88,93.61},{66.98,144.61},{67.21,133.07},
  {37.90,81.47},{68.35,136.90},{69.28,140.78},{78.26,143.36},
  {28.73,69.02},{48.85,91.09},{ 6.11,61.37},{69.24,156.76},
  {58.32,123.43},{13.23,59.97},{32.85,74.58},{48.15,120.84},
  {74.60,145.21},{46.64,104.61},{63.37,120.76},{13.36,59.46},
  {69.89,142.17},{67.89,136.10},{49.22,99.19},{73.16,143.29},
  {47.79,130.64},{41.71,94.63},{93.46,171.77},{99.74,185.80},
  {58.15,112.90},{24.90,82.06},{17.53,58.51},{34.06,80.58},
  {51.11,115.72},{19.12,64.68},{29.05,80.50},{30.71,91.87},
  {20.00,77.43},{38.82,97.86},{25.56,71.28},{24.69,51.78},
  {15.15,52.31},{89.92,178.26},{97.21,171.26},{54.16,134.74},
  {84.67,149.81},{74.93,123.09},{ 5.26,24.90},{99.04,183.04},
  {89.69,180.72},{ 9.57,59.78},{27.52,86.77},{ 7.79,63.47},
  {86.70,159.99},{12.54,49.44},{65.80,139.16},{60.68,99.45},
  {37.01,99.10},{65.32,128.72},{79.27,139.94},{13.48,59.51},
  {16.15,65.81},{ 5.50,56.27},{21.44,61.06},{17.95,80.39},
  {22.99,69.66},{78.04,139.81},{ 8.19,45.53},{53.04,114.50},
  {22.03,55.53},{71.11,134.99},{12.41,60.57},{47.53,107.37},
  { 0.20,27.63},{ 3.31,26.26},{59.81,132.51},{50.17,104.10},
  {84.25,141.14},{91.89,171.66},{14.95,62.25},{ 9.00,61.06},
  {29.68,75.94},{98.55,169.15},{63.95,127.72},{41.36,82.03},
  {92.20,168.84},{71.55,142.74},{89.17,168.56},{36.19,84.82},
  { 5.83,58.93},{32.71,82.95},{13.63,72.12},{20.78,69.59},
  {96.66,156.89},{40.74,93.92},{12.50,64.55},{91.70,165.65},
  {45.68,89.74},{10.70,52.42},{80.60,159.09},{46.91,99.34},
  {42.30,97.16},{34.03,85.62},{68.84,132.20},{94.47,166.73},
  { 6.57,23.33},{88.09,172.72},{10.29,44.01},{16.28,64.39},
  {40.21,82.53},{42.50,101.48},{85.18,145.73},{88.49,176.79},
  {23.93,69.17},{21.58,71.42},{43.56,101.34},{18.85,72.03},
  { 4.01,20.86},{58.74,130.89},{ 0.55,42.23},{64.01,138.48},
  {86.32,164.34},{ 4.01,62.96},{71.65,145.59},{59.98,128.80},
  {47.29,107.25},{52.80,112.62},{73.48,143.42},{60.71,105.76},
  {14.39,46.36},{91.65,166.65},{68.70,134.37},{17.20,63.05},
  {49.86,111.33},{15.66,66.77},{13.85,55.13},{11.74,62.94},
  {46.11,92.86},{90.43,144.43},{12.80,46.53},{ 8.49,48.78},
  {92.34,176.52},{77.18,145.53},{18.95,72.13},{25.16,77.45},
  {79.17,156.72},{94.54,168.51},{12.56,52.73},{31.32,80.71},
  {83.67,145.98},{69.02,141.65},{16.67,51.28},{43.22,108.69},
  { 2.77,44.28},{28.19,92.70},{85.57,161.86},{16.23,62.41},
  { 7.59,70.56},{36.61,85.26},{31.17,83.60},{77.49,151.10},
  {12.82,38.79},{30.11,81.59},{50.07,122.10},{74.50,144.63},
  {94.48,175.21},{82.49,146.39},{47.18,90.69},{19.81,68.22},
  {67.87,135.07},{86.53,158.63},{ 4.02,67.70},{79.22,163.68},
  {18.63,65.68},{93.39,170.96},{95.97,163.34},{75.47,121.35},
  { 0.78,37.11},{ 9.53,50.40},{39.13,110.03},{95.69,168.67},
  {27.61,84.96},{47.10,120.52},{96.66,178.29},{88.15,179.79},
  {54.08,127.28},{98.67,173.36},{28.33,79.71},{ 3.98,31.32},
  {98.84,179.12},{22.71,70.55},{ 2.25,35.21},{32.51,72.10},
  {61.33,121.66},{70.04,137.97},{47.57,129.83},{15.27,63.70},
  {67.47,148.68},{90.29,162.66},{ 5.58,56.85},{26.24,75.05},
  {97.20,190.42},{97.93,174.98},{72.40,139.24},{36.57,100.59},
  { 9.55,69.48},{28.55,80.48},{23.97,69.20},{40.40,94.65},
  {93.43,169.59},{56.99,101.50},{29.82,78.34},{63.85,105.18},
  {36.57,93.67},{29.99,100.46},{48.09,99.60},{17.13,85.66},
  {42.67,102.26},{26.34,76.52},{ 9.81,48.84},{35.70,76.14},
  {89.40,153.75},{97.80,177.01},{27.89,69.25},{46.43,113.97},
  {21.64,62.84},{72.79,131.04},{86.23,150.89},{57.53,122.30},
  {36.87,91.32},{13.15,50.63},{81.13,165.31},{29.36,108.50},
  {25.65,81.54},{21.91,58.02},{60.06,128.34},{53.90,116.97},
  {37.20,91.22},{ 2.75,54.02},{78.84,143.43},{78.42,129.08},
  {30.30,91.45},{ 6.19,51.64},{15.94,75.49},{49.50,107.77},
  {48.58,103.97},{42.05,114.46},{98.55,169.49},{23.59,77.96},
  { 4.95,31.04},{51.61,122.22},{89.57,166.43},{97.29,183.00},
  {67.36,143.50},{70.70,143.79},{ 7.09,61.57},{ 4.55,35.73},
  { 3.12,26.08},{27.61,71.71},{17.87,65.37},{73.82,148.35},
  {71.86,152.17},{39.75,97.64},{11.52,51.38},{84.82,150.22},
  {33.13,77.12},{34.83,83.95},{53.84,105.93},{85.86,161.20},
  {80.36,135.81},{29.48,66.91},{33.44,84.75},{27.94,89.60},
  {61.89,130.52},{15.65,50.50},{66.84,126.11},{61.89,124.02},
  {30.64,82.56},{63.67,113.67},{93.79,175.50},{89.78,180.21},
  {49.60,106.06},{78.60,152.09},{88.82,171.67},{ 4.49,41.76},
  {12.41,62.47},{57.54,122.63},{42.00,96.54},{15.89,62.16},
  {18.09,43.62},{98.19,177.35},{49.84,105.13},{59.38,128.63},
  {55.34,118.20},{60.21,125.47},{31.34,69.51},{79.20,139.77},
  {26.37,81.64},{45.32,72.27},{91.13,173.22},{91.36,169.43},
  {65.10,128.76},{24.33,59.90},{39.37,93.39},{88.88,156.65},
  {66.86,146.50},{73.40,126.02},{14.09,64.93},{87.34,173.83},
  {18.26,68.89},{92.26,160.92},{77.91,157.56},{52.89,98.98},
  {38.14,109.31},{41.50,96.53},{26.81,89.59},{47.42,103.41},
  {68.58,132.42},{60.29,126.09},{64.99,125.45},{76.35,144.33},
  {11.69,57.61},{28.16,72.44},{23.94,72.18},{95.67,182.61},
  {59.17,118.32},{35.19,83.30},{19.53,74.53},{45.16,96.72},
  {66.63,128.79},{96.13,182.09},{65.31,126.98},{33.27,102.77},
  { 3.65,39.52},{19.26,74.36},{32.61,70.26},{37.77,82.99},
  { 1.77,32.37},{87.50,167.27},{90.60,158.93},{86.81,154.12},
  {23.83,77.55},{97.47,166.55},{83.99,167.80},{44.51,104.49},
  {86.46,168.85},{75.17,142.50},{83.71,173.31},{92.83,162.93}
};
```
Illustration 18: Linear Regression (original program)

Mean Running time of original program:



no of run time	Taken time(s)
1	0.09509
2	0.09477
3	0.09526
4	0.09376
5	0.09369
6	0.09381
7	0.09406
8	0.09473
9	0.09488
10	0.09756
Mean running time	0.094761

Illustration 19: Linear Regression (original program)

e) During each iteration of the program, eight values for m and c are evaluated to measure the resulting error. Create a modified version of the program that performs each of the evaluations on a different thread.

Answer:

The modified program file is available in Folder POSIX Threads-> Linear Regression with file named as 34_multithread_T.c

- Structure is created for striding technique:

typedef struc tpoint_t{double x; double y;} point_t;

- Pthread function is created: to calculate rms error

void *regression_thread(void *args)

- 8 Pthread is created in a loop:

for(t=0; t<8; t++)

{pthread_create(&threads[t], NULL, regression_thread,&t);

pthread_join(threads[t], NULL); printf("Best m,c is %lf,%lf with error %lf in direction %d\n",

dm[best_error_i], dc[best_error_i], best_error, best_error_i);

}

Source code for a linear regression thread:

```
#include <stdio.h>
#include <math.h>
#include <pthread.h>

/*************************************************************************
 * To compile:
 *   cc -o lr_coursework 34_multithread_T.c -lm -pthread
 *
 * To run:
 *   ./lr_coursework


*************************************************************************
***/
```

```
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};


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

int n_data = 1000;
point_t data[];

double 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);
}

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);
}
```

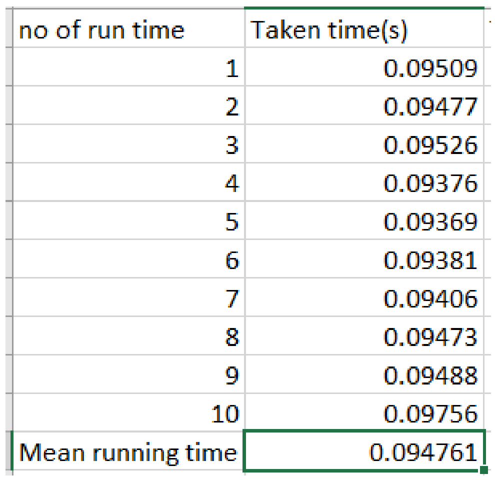
```
void *linear_regression_thread(void *args){

  int *a = args;
  int i = *a;
  printf("\n i in thread fun=%d", i);
  dm[i] = bm + (om[i] * step);
      dc[i] = bc + (oc[i] * step);

  e[i] = rms_error(dm[i], dc[i]);
  if(e[i] < best_error) {
  best_error = e[i];
  best_error_i = i;
  pthread_exit(NULL);
  }
}

int main() {
  struct timespec start, finish;
  long long int time_elapsed;
  clock_gettime(CLOCK_MONOTONIC, &start);

  int i;
  pthread_t p_threads[8];

  be = rms_error(bm, bc);

  while(!minimum_found) {
    for(i=0;i<8;i++) {
      pthread_create(&p_threads[i], NULL, (void*)linear_regression_thread,
&i);
      pthread_join(p_threads[i], NULL);
    }

    printf("best m,c is %lf,%lf with error %lf in direction %d\n",
      dm[best_error_i], dc[best_error_i], best_error, best_error_i);

    if(best_error < be) {
      be = best_error;
      bm = dm[best_error_i];
      bc = dc[best_error_i];
    } else {
      minimum_found = 1;
    }
  }
  printf("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);

  clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
(time_elapsed/1.0e9));

  pthread_exit(NULL);
  return 0;
```

```
}
point_t data[] = {

   {65.11,126.40},{76.79,149.00},{76.93,162.00},{65.24,113.46},
  {78.54,145.93},{84.60,161.77},{85.60,162.58},{82.32,152.21},
  {78.17,144.80},{69.47,142.78},{82.72,156.05},{11.56,52.20},
  {66.15,122.01},{75.13,145.75},{ 8.11,36.01},{71.58,150.44},
  {23.30,70.06},{42.59,86.42},{39.11,76.86},{ 8.77,36.29},
  {83.41,152.32},{ 3.44,36.86},{72.15,126.11},{66.29,129.15},
  {28.93,92.24},{91.62,172.01},{ 0.39,40.02},{55.24,104.88},
  {44.96,90.98},{89.66,170.29},{29.39,86.66},{56.19,109.96},
  {79.43,153.15},{54.27,110.73},{ 9.28,54.91},{31.16,76.74},
  {20.00,49.29},{67.25,122.68},{95.64,182.45},{66.03,128.30},
  {36.60,94.60},{83.93,120.16},{22.67,76.02},{81.17,164.59},
  {84.70,147.15},{34.58,87.81},{ 0.26,39.25},{82.07,149.43},
  { 2.63,41.39},{ 1.74, 7.95},{70.98,133.50},{16.65,49.48},
  {27.85,61.85},{55.84,105.63},{81.77,153.60},{19.81,61.26},
  {28.19,97.35},{ 2.62,32.52},{60.42,123.43},{53.67,118.83},
  {92.67,163.43},{ 4.09,30.06},{31.35,78.55},{54.79,103.97},
  {89.15,163.38},{20.35,66.02},{28.55,88.62},{11.66,58.29},
  {89.90,154.40},{ 0.14,51.92},{ 4.75,37.69},{53.83,108.99},
  {62.17,127.68},{79.10,133.64},{24.19,68.78},{51.41,100.86},
  {44.52,92.93},{23.02,66.51},{98.60,181.12},{ 6.05,48.82},
  {62.79,147.70},{ 5.06,50.58},{85.40,155.28},{12.33,60.17},
  {49.62,118.33},{ 9.03,48.29},{45.21,88.73},{28.22,55.37},
  {91.32,165.67},{ 6.74,44.19},{46.03,93.83},{69.75,139.69},
  { 2.15,40.31},{95.82,160.20},{ 6.64,54.91},{75.25,148.74},
  {39.64,68.97},{ 5.55,66.26},{90.53,155.37},{39.95,91.42},
  {68.89,132.98},{33.52,78.37},{15.84,38.51},{72.73,139.50},
  {21.54,73.78},{ 4.64,47.34},{66.57,132.87},{27.38,71.86},
  {93.83,181.33},{75.83,161.75},{26.47,56.70},{84.23,151.43},
  { 0.43,43.29},{88.50,160.27},{66.15,129.59},{78.31,141.68},
  {36.90,101.61},{71.78,139.52},{90.37,173.79},{ 0.58,45.98},
  {67.63,131.85},{57.43,100.37},{88.43,161.15},{74.83,132.98},
  {29.31,54.66},{79.06,146.78},{54.41,120.80},{51.76,108.96},
  {11.80,65.51},{38.19,90.48},{18.40,71.77},{76.29,148.07},
  {75.30,135.15},{59.56,126.34},{32.71,86.25},{42.35,116.15},
  { 4.85,38.50},{ 3.14,50.60},{48.27,90.59},{34.96,88.02},
  {10.03,50.01},{ 5.51,40.83},{68.32,136.16},{74.87,134.02},
  { 1.56,47.50},{19.52,72.68},{ 9.10,52.12},{50.79,102.10},
  {53.11,105.38},{94.93,174.68},{16.03,44.26},{13.26,49.58},
  { 3.24,46.86},{77.38,158.65},{21.57,62.81},{41.63,89.86},
  {13.55,59.72},{25.43,71.35},{86.73,166.79},{77.15,149.52},
  {26.47,64.94},{48.65,92.62},{33.66,75.10},{25.20,63.45},
  {25.46,86.18},{70.52,147.39},{98.32,175.47},{23.09,62.68},
  {48.90,118.74},{69.07,141.45},{50.54,132.25},{55.80,119.88},
  {25.65,92.01},{54.39,112.81},{79.86,165.28},{95.98,154.86},
  {48.14,108.06},{36.33,88.43},{ 6.34,35.96},{86.04,151.77},
  {57.03,116.32},{97.95,180.12},{29.66,73.83},{12.52,35.04},
  {43.93,83.56},{33.63,78.69},{64.00,128.13},{14.49,50.14},
  {49.66,112.89},{82.54,162.20},{81.92,143.19},{28.07,78.90},
  {14.26,47.92},{23.97,63.31},{27.69,73.01},{78.13,119.54},
  {34.43,82.48},{66.13,123.89},{61.84,135.81},{17.03,57.30},
  { 5.61,52.51},{34.44,88.49},{17.81,81.52},{34.26,79.71},
  {93.17,161.29},{ 8.10,39.44},{93.51,158.23},{61.48,133.51},
  {27.22,71.93},{17.11,50.22},{27.73,81.68},{16.07,61.27},
  {63.81,122.63},{ 0.27,36.83},{62.21,120.74},{42.36,85.01},
  {60.61,143.23},{68.59,121.10},{28.48,68.27},{23.39,71.50},
  {93.40,162.60},{50.72,114.87},{24.53,80.83},{92.00,160.38},
  {79.29,175.12},{28.84,78.42},{13.79,44.14},{23.18,62.24},
  {69.07,122.23},{41.93,120.63},{62.32,125.07},{72.39,136.08},
  {41.86,92.41},{ 2.35,21.43},{56.14,133.02},{33.91,90.07},
  {13.68,76.01},{14.55,71.51},{73.79,152.07},{33.47,97.28},
  {31.12,65.68},{ 4.33,41.19},{22.94,58.10},{85.12,160.37},
  {80.26,154.39},{37.01,78.54},{ 6.94,38.10},{ 7.83,60.51},
  {42.44,90.12},{26.69,91.63},{99.36,184.47},{ 9.33,55.05},
  {16.87,63.97},{32.41,80.49},{36.34,77.15},{59.91,122.62},
  { 7.35,34.97},{99.21,183.69},{34.07,97.06},{43.85,102.96},
  {55.20,111.99},{ 3.95,41.47},{26.71,82.57},{16.69,64.61},
  {38.32,96.85},{76.67,136.21},{86.22,175.61},{35.18,71.39},
  {57.39,117.85},{72.12,139.30},{90.19,173.32},{97.26,163.25},
  {82.08,135.04},{40.69,96.78},{25.49,75.76},{83.38,149.39},
  {63.64,135.54},{90.52,166.25},{79.96,154.68},{45.70,107.42},
  {15.54,61.07},{97.10,170.63},{41.10,87.33},{35.86,74.41},
  {57.22,120.65},{16.28,64.80},{46.33,97.53},{31.84,83.82},
  {90.15,177.85},{13.39,77.75},{ 8.25,26.83},{91.74,155.44},
  {11.65,61.09},{26.30,82.75},{61.72,128.34},{76.94,152.59},
  {26.70,81.25},{ 6.11,39.57},{97.46,172.22},{13.50,43.37},
  {16.46,54.81},{44.36,84.30},{45.83,105.17},{41.47,105.60},
  {31.72,82.66},{58.79,113.04},{95.35,168.80},{27.91,73.77},
  {61.28,126.49},{ 1.18,32.24},{ 4.17,41.67},{79.08,142.98},
  { 4.80,37.58},{94.98,160.79},{37.47,80.15},{ 6.82,53.58},
  {57.54,118.51},{73.31,139.05},{91.40,166.67},{98.07,162.54},
  { 3.87,41.53},{63.71,130.41},{75.78,137.34},{56.32,122.22},
  { 8.03,63.52},{22.60,60.09},{94.56,158.12},{16.10,72.60},
  {82.22,155.28},{57.63,114.94},{55.26,118.38},{73.52,126.54},
  {59.13,123.52},{81.11,167.56},{68.73,123.16},{43.78,122.00},
  {27.48,70.82},{43.92,110.09},{ 7.04,44.70},{91.03,147.13},
  {44.55,114.71},{68.40,122.77},{ 6.31,55.15},{12.03,51.80},
  {26.62,77.09},{12.90,60.85},{41.47,115.00},{75.98,156.05},
  {62.84,118.68},{ 3.19,54.74},{74.93,132.72},{89.37,170.57},
  {57.12,114.25},{63.07,104.90},{60.20,129.41},{36.04,82.56},
  {66.43,133.16},{ 7.01,45.98},{87.68,162.25},{36.24,86.35},
  {60.38,140.39},{ 3.56,41.80},{65.74,138.56},{16.34,61.32},
  {34.00,105.97},{77.42,132.46},{61.79,135.32},{61.13,116.51},
  {90.22,159.21},{68.78,137.53},{48.35,110.41},{58.48,110.06},
  { 0.04,45.54},{87.14,152.92},{73.42,154.90},{48.73,105.35},
  {36.26,97.55},{50.34,107.46},{95.65,162.87},{11.76,50.19},
  {12.83,58.44},{ 7.59,65.35},{51.44,109.31},{15.39,65.30},
  {83.69,147.00},{75.86,139.91},{25.87,87.89},{ 0.79,40.68},
  { 4.87,62.14},{64.71,126.73},{60.94,111.46},{82.18,142.90},
  {21.67,55.08},{33.20,93.76},{11.93,64.07},{10.59,39.64},
  {20.90,63.85},{21.47,81.20},{15.02,80.95},{67.04,112.88},
  { 0.78,34.00},{24.49,77.44},{ 0.49,33.31},{70.88,138.01},
  {33.07,81.36},{74.11,139.95},{ 1.26,40.24},{26.68,81.03},
  {81.37,155.39},{89.28,163.35},{82.92,150.44},{88.94,166.64},
  {14.35,63.72},{58.92,119.69},{47.88,100.06},{73.51,145.81},
  {21.11,79.13},{33.98,87.52},{87.88,158.32},{24.47,79.36},
  {34.00,92.04},{42.25,105.99},{20.75,64.71},{65.45,131.84},
  {10.51,38.23},{36.26,87.53},{70.72,129.61},{62.87,129.51},
  {88.93,170.73},{50.72,124.30},{52.28,104.85},{ 1.38,36.72},
  {93.31,171.99},{56.94,135.14},{99.80,174.15},{ 2.74,46.10},
  {42.39,81.82},{42.07,102.65},{ 3.06,34.94},{69.46,129.37},
  {43.33,104.41},{69.47,129.06},{68.70,134.00},{95.28,179.59},
  {66.66,132.99},{14.40,70.02},{39.22,77.48},{ 5.87,32.85},
  {71.70,144.19},{69.40,140.87},{51.33,113.42},{38.58,97.94},
  {70.18,129.30},{28.53,76.70},{11.77,52.53},{16.26,69.96},
  {86.39,161.92},{14.35,60.74},{17.32,53.69},{60.10,117.59},
  {55.86,117.63},{ 9.26,58.48},{48.28,100.74},{79.81,149.90},
  {59.38,105.17},{72.31,119.55},{41.23,91.88},{70.20,136.75},
  {82.58,158.00},{72.29,130.76},{10.80,60.44},{81.60,167.47},
  {57.21,120.91},{83.97,150.98},{78.97,152.79},{78.71,146.61},
  {98.28,172.82},{39.89,87.34},{92.46,169.18},{29.94,69.32},
  {64.78,127.52},{52.32,113.01},{29.54,78.81},{46.04,97.77},
  {97.71,155.81},{90.08,172.72},{59.58,117.24},{74.61,149.80},
  {64.20,116.45},{42.14,102.85},{44.27,107.95},{97.48,174.75},
  {84.52,164.71},{19.46,51.01},{87.05,166.73},{28.47,61.63},
  { 0.59,40.82},{49.83,100.11},{25.05,62.68},{20.32,62.69},
  {56.64,126.50},{15.41,59.79},{98.36,173.55},{57.73,97.55},
  {29.88,86.10},{26.44,65.10},{92.31,174.76},{10.74,49.39},
  {88.32,163.82},{71.67,156.47},{94.40,181.01},{16.04,55.25},
  {76.78,141.43},{81.76,146.31},{81.41,144.40},{ 7.59,65.42},
  {29.83,93.19},{71.85,128.96},{17.45,45.91},{66.78,148.29},
  {87.12,149.00},{55.37,115.72},{ 8.34,69.56},{10.30,58.08},
  {26.49,79.04},{33.65,83.88},{82.84,146.92},{74.19,132.28},
  {47.38,105.49},{95.30,172.52},{ 8.60,54.66},{38.14,69.96},
  {14.06,65.00},{27.26,74.14},{31.28,82.60},{24.83,97.76},
  {53.38,101.53},{26.88,87.40},{77.79,153.42},{ 4.60,53.07},
  {36.70,76.53},{33.96,93.76},{64.04,116.47},{73.85,156.81},
  {31.88,85.18},{10.44,62.68},{41.55,99.48},{31.20,94.01},
  {69.63,137.46},{30.90,92.46},{54.24,103.71},{82.12,149.86},
  {57.43,119.86},{16.83,61.60},{38.45,74.99},{ 9.38,60.12},
  {18.91,53.31},{65.28,115.98},{52.45,119.66},{ 4.88,52.30},
  {49.92,95.81},{60.44,126.35},{25.07,83.25},{58.21,108.57},
  {28.81,94.01},{44.94,99.55},{35.75,79.36},{95.26,164.51},
  {35.14,113.38},{ 6.95,51.77},{37.56,81.49},{27.79,72.55},
  {68.04,129.74},{19.46,82.90},{13.49,40.02},{40.81,92.01},
  {44.13,86.46},{90.16,178.62},{82.34,153.55},{92.32,165.64},
  {78.20,166.78},{24.76,66.00},{91.00,175.56},{85.94,172.81},
  {98.74,178.91},{ 7.47,34.02},{37.28,85.95},{ 8.94,54.67},
  {31.78,85.85},{31.71,82.87},{44.29,96.83},{21.85,68.96},
  {15.20,48.69},{ 9.51,58.55},{14.53,53.46},{87.25,166.09},
  {35.25,79.77},{45.43,106.79},{16.24,69.34},{61.36,142.83},
  {99.33,176.18},{ 2.66,42.92},{42.69,105.85},{69.04,124.32},
  {62.77,125.38},{87.76,149.94},{68.38,124.70},{44.95,109.62},
  { 8.36,63.51},{29.47,75.02},{42.49,87.92},{29.05,95.14},
  { 1.36,43.70},{60.36,102.16},{23.57,54.88},{30.84,80.74},
  {10.19,42.03},{97.59,177.44},{36.08,89.31},{21.74,45.86},
  {58.56,113.61},{34.10,92.54},{87.76,174.79},{43.42,107.45},
  {55.01,110.06},{45.87,119.35},{21.24,61.64},{ 0.63,23.13},
  {44.94,99.54},{ 5.22,47.01},{ 1.71,42.19},{92.32,159.09},
  {28.15,76.89},{77.98,128.92},{40.11,84.47},{80.44,144.10},
  {21.62,80.78},{27.18,70.12},{80.83,148.92},{65.54,132.52},
  {69.13,124.43},{26.54,59.95},{ 0.13,36.97},{24.07,70.64},
  {27.58,70.42},{45.07,121.14},{11.82,46.41},{81.39,156.60},
  {49.46,95.96},{56.25,93.87},{92.93,167.21},{85.35,169.34},
  {32.46,93.55},{37.88,93.61},{66.98,144.61},{67.21,133.07},
  {37.90,81.47},{68.35,136.90},{69.28,140.78},{78.26,143.36},
  {28.73,69.02},{48.85,91.09},{ 6.11,61.37},{69.24,156.76},
  {58.32,123.43},{13.23,59.97},{32.85,74.58},{48.15,120.84},
  {74.60,145.21},{46.64,104.61},{63.37,120.76},{13.36,59.46},
  {69.89,142.17},{67.89,136.10},{49.22,99.19},{73.16,143.29},
  {47.79,130.64},{41.71,94.63},{93.46,171.77},{99.74,185.80},
  {58.15,112.90},{24.90,82.06},{17.53,58.51},{34.06,80.58},
  {51.11,115.72},{19.12,64.68},{29.05,80.50},{30.71,91.87},
  {20.00,77.43},{38.82,97.86},{25.56,71.28},{24.69,51.78},
  {15.15,52.31},{89.92,178.26},{97.21,171.26},{54.16,134.74},
  {84.67,149.81},{74.93,123.09},{ 5.26,24.90},{99.04,183.04},
  {89.69,180.72},{ 9.57,59.78},{27.52,86.77},{ 7.79,63.47},
  {86.70,159.99},{12.54,49.44},{65.80,139.16},{60.68,99.45},
  {37.01,99.10},{65.32,128.72},{79.27,139.94},{13.48,59.51},
  {16.15,65.81},{ 5.50,56.27},{21.44,61.06},{17.95,80.39},
  {22.99,69.66},{78.04,139.81},{ 8.19,45.53},{53.04,114.50},
  {22.03,55.53},{71.11,134.99},{12.41,60.57},{47.53,107.37},
  { 0.20,27.63},{ 3.31,26.26},{59.81,132.51},{50.17,104.10},
  {84.25,141.14},{91.89,171.66},{14.95,62.25},{ 9.00,61.06},
  {29.68,75.94},{98.55,169.15},{63.95,127.72},{41.36,82.03},
  {92.20,168.84},{71.55,142.74},{89.17,168.56},{36.19,84.82},
  { 5.83,58.93},{32.71,82.95},{13.63,72.12},{20.78,69.59},
  {96.66,156.89},{40.74,93.92},{12.50,64.55},{91.70,165.65},
  {45.68,89.74},{10.70,52.42},{80.60,159.09},{46.91,99.34},
  {42.30,97.16},{34.03,85.62},{68.84,132.20},{94.47,166.73},
  { 6.57,23.33},{88.09,172.72},{10.29,44.01},{16.28,64.39},
  {40.21,82.53},{42.50,101.48},{85.18,145.73},{88.49,176.79},
  {23.93,69.17},{21.58,71.42},{43.56,101.34},{18.85,72.03},
  { 4.01,20.86},{58.74,130.89},{ 0.55,42.23},{64.01,138.48},
  {86.32,164.34},{ 4.01,62.96},{71.65,145.59},{59.98,128.80},
  {47.29,107.25},{52.80,112.62},{73.48,143.42},{60.71,105.76},
  {14.39,46.36},{91.65,166.65},{68.70,134.37},{17.20,63.05},
  {49.86,111.33},{15.66,66.77},{13.85,55.13},{11.74,62.94},
  {46.11,92.86},{90.43,144.43},{12.80,46.53},{ 8.49,48.78},
  {92.34,176.52},{77.18,145.53},{18.95,72.13},{25.16,77.45},
  {79.17,156.72},{94.54,168.51},{12.56,52.73},{31.32,80.71},
  {83.67,145.98},{69.02,141.65},{16.67,51.28},{43.22,108.69},
  { 2.77,44.28},{28.19,92.70},{85.57,161.86},{16.23,62.41},
  { 7.59,70.56},{36.61,85.26},{31.17,83.60},{77.49,151.10},
  {12.82,38.79},{30.11,81.59},{50.07,122.10},{74.50,144.63},
  {94.48,175.21},{82.49,146.39},{47.18,90.69},{19.81,68.22},
  {67.87,135.07},{86.53,158.63},{ 4.02,67.70},{79.22,163.68},
  {18.63,65.68},{93.39,170.96},{95.97,163.34},{75.47,121.35},
  { 0.78,37.11},{ 9.53,50.40},{39.13,110.03},{95.69,168.67},
  {27.61,84.96},{47.10,120.52},{96.66,178.29},{88.15,179.79},
  {54.08,127.28},{98.67,173.36},{28.33,79.71},{ 3.98,31.32},
  {98.84,179.12},{22.71,70.55},{ 2.25,35.21},{32.51,72.10},
  {61.33,121.66},{70.04,137.97},{47.57,129.83},{15.27,63.70},
  {67.47,148.68},{90.29,162.66},{ 5.58,56.85},{26.24,75.05},
  {97.20,190.42},{97.93,174.98},{72.40,139.24},{36.57,100.59},
  { 9.55,69.48},{28.55,80.48},{23.97,69.20},{40.40,94.65},
  {93.43,169.59},{56.99,101.50},{29.82,78.34},{63.85,105.18},
  {36.57,93.67},{29.99,100.46},{48.09,99.60},{17.13,85.66},
  {42.67,102.26},{26.34,76.52},{ 9.81,48.84},{35.70,76.14},
  {89.40,153.75},{97.80,177.01},{27.89,69.25},{46.43,113.97},
  {21.64,62.84},{72.79,131.04},{86.23,150.89},{57.53,122.30},
  {36.87,91.32},{13.15,50.63},{81.13,165.31},{29.36,108.50},
  {25.65,81.54},{21.91,58.02},{60.06,128.34},{53.90,116.97},
  {37.20,91.22},{ 2.75,54.02},{78.84,143.43},{78.42,129.08},
  {30.30,91.45},{ 6.19,51.64},{15.94,75.49},{49.50,107.77},
  {48.58,103.97},{42.05,114.46},{98.55,169.49},{23.59,77.96},
  { 4.95,31.04},{51.61,122.22},{89.57,166.43},{97.29,183.00},
  {67.36,143.50},{70.70,143.79},{ 7.09,61.57},{ 4.55,35.73},
  { 3.12,26.08},{27.61,71.71},{17.87,65.37},{73.82,148.35},
  {71.86,152.17},{39.75,97.64},{11.52,51.38},{84.82,150.22},
  {33.13,77.12},{34.83,83.95},{53.84,105.93},{85.86,161.20},
  {80.36,135.81},{29.48,66.91},{33.44,84.75},{27.94,89.60},
  {61.89,130.52},{15.65,50.50},{66.84,126.11},{61.89,124.02},
  {30.64,82.56},{63.67,113.67},{93.79,175.50},{89.78,180.21},
  {49.60,106.06},{78.60,152.09},{88.82,171.67},{ 4.49,41.76},
  {12.41,62.47},{57.54,122.63},{42.00,96.54},{15.89,62.16},
  {18.09,43.62},{98.19,177.35},{49.84,105.13},{59.38,128.63},
  {55.34,118.20},{60.21,125.47},{31.34,69.51},{79.20,139.77},
  {26.37,81.64},{45.32,72.27},{91.13,173.22},{91.36,169.43},
  {65.10,128.76},{24.33,59.90},{39.37,93.39},{88.88,156.65},
  {66.86,146.50},{73.40,126.02},{14.09,64.93},{87.34,173.83},
  {18.26,68.89},{92.26,160.92},{77.91,157.56},{52.89,98.98},
  {38.14,109.31},{41.50,96.53},{26.81,89.59},{47.42,103.41},
  {68.58,132.42},{60.29,126.09},{64.99,125.45},{76.35,144.33},
  {11.69,57.61},{28.16,72.44},{23.94,72.18},{95.67,182.61},
  {59.17,118.32},{35.19,83.30},{19.53,74.53},{45.16,96.72},
  {66.63,128.79},{96.13,182.09},{65.31,126.98},{33.27,102.77},
  { 3.65,39.52},{19.26,74.36},{32.61,70.26},{37.77,82.99},
  { 1.77,32.37},{87.50,167.27},{90.60,158.93},{86.81,154.12},
  {23.83,77.55},{97.47,166.55},{83.99,167.80},{44.51,104.49},
  {86.46,168.85},{75.17,142.50},{83.71,173.31},{92.83,162.93}
};
```
Illustration 20: Linear Regression (Thread Program)

f) Calculate the mean running time of the multithread version of the program and use to make comments on the relative performance of the 2 versions.

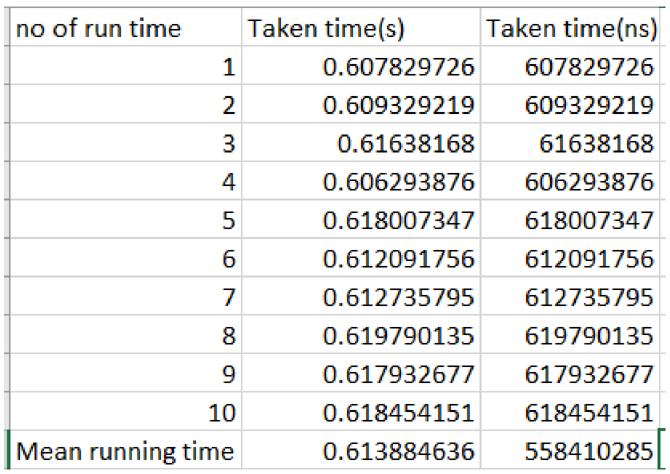
Answer:

Mean running time:



no of run time	Taken time(s)
1	0.09509
2	0.09477
3	0.09526
4	0.09376
5	0.09369
6	0.09381
7	0.09406
8	0.09473
9	0.09488
10	0.09756
Mean running time	0.094761

Illustration 21: Linear Regression (original mean time)



no of run time	Taken time(s)	Taken time(ns)
1	0.607829726	607829726
2	0.609329219	609329219
3	0.61638168	61638168
4	0.606293876	606293876
5	0.618007347	618007347
6	0.612091756	612091756
7	0.612735795	612735795
8	0.619790135	619790135
9	0.617932677	617932677
10	0.618454151	618454151
Mean running time	0.613884636	558410285

Illustration 22: Linear Regression with posix thread

2.3.1 Posix Linear Regression Code Analysis:

The main aim of the code is to find out the minimum m and c value with best error using CUDA tooklit. M and c are the slope of line and intercept of the staright line. Slope deals with direction and lines steepness where intercepts are points where it crosses the axis. There are intercepts x-intercept and y- inetcept. We tried to findout the values of m and c with optimal root mean square error. Two time-spec start and finish and finish along with elapsed is declared in declaration phase to access the start time , finish time and the time difference. Clock_get time function is used to get the time from the system which takes two parameters CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time. Time elapsed is calculated by subtracting finish time and start time after getting the both time.

3. CUDA:

3.1 Password Cracking:

a) Create another version of the two-initials-two-digits password cracker. It should use separate CUDA thread to process each possible pair of initials, i.e. thread 0 should process AA through to thread675 processing ZZ.

Answer:

The password with two-initials-four-digits password are mentioned below.

i. RA27

ii. HT82

iii. FS83

iv.BV78

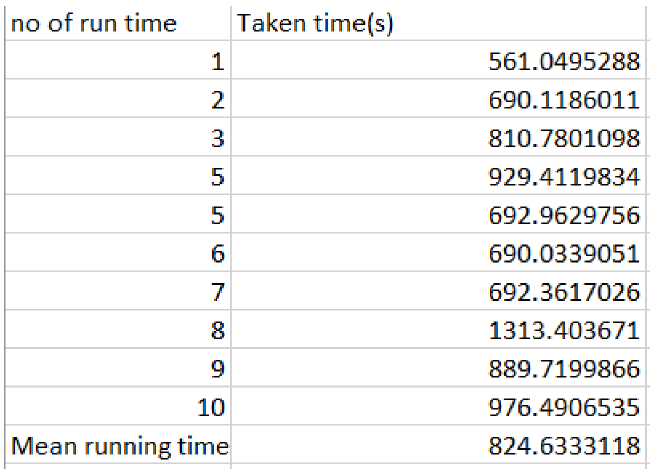
Source code:

```
#include <stdio.h>
#include <cuda_runtime_api.h>
#include <time.h>
/***********************************************************************
  Compile with:
    nvcc -o passwordcrackwith2initial_cuda
passwordcrackwith2initial_cuda.cu

  Run with:

    ./passwordcrackwith2initial_cuda

************************************************************************/
```

```
__device__ int is_a_match(char *attempt) {
        char plain_password1[] = "RA27";
        char plain_password2[] = "HT82";
        char plain_password3[] = "FS83";
        char plain_password4[] = "BV78";


        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("Password: %s\n",plain_password1);
                        break;
                }

                a++;
                p1++;
        }

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

                b++;
                p2++;
        }

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

                c++;
                p3++;
        }

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

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

}

__global__ void  kernel() {
        char i1,i2,i3,i4;

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

        int i = blockIdx.x+65;
        int j = threadIdx.x+65;
        char firstMatch = i;
        char secondMatch = j;

        password[0] = firstMatch;
        password[1] = secondMatch;
        for(i1='0'; i1<='9'; i1++){
                for(i2='0'; i2<='9'; i2++){
                        for(i3='0'; i3<='9'; i3++){
                                for(i4='0'; i4<='9'; i4++){
                                        password[2] = i1;
                                        password[3] = i2;
                                        password[4] = i3;
                                        password[5] = i4;
                                        if(is_a_match(password)) {
                                        }
                                        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);
}
```

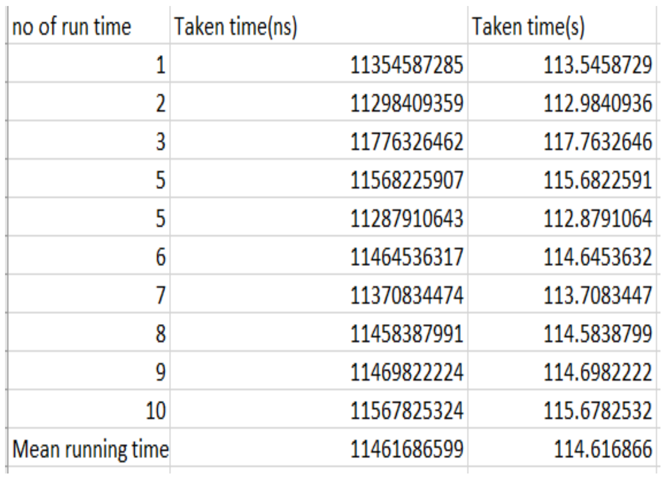
```
int main() {

        struct  timespec start, finish;
        long long int time_elapsed;
        clock_gettime(CLOCK_MONOTONIC, &start);

        kernel <<<26,26>>>();
        cudaThreadSynchronize();

        clock_gettime(CLOCK_MONOTONIC, &finish);
        time_difference(&start, &finish, &time_elapsed);
        printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
(time_elapsed/1.0e9));

        return 0;
}
```
Illustration 23: Cuda password cracking (2initial 2 digits)

b) The comparison of mean running time of the CUDA version with the original and multithread versions.



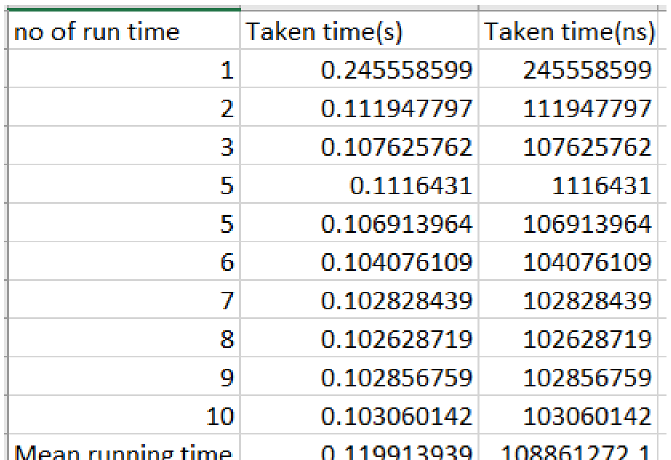
no of run time	Taken time(s)
1	561.0495288
2	690.1186011
3	810.7801098
5	929.4119834
5	692.9629756
6	690.0339051
7	692.3617026
8	1313.403671
9	889.7199866
10	976.4906535
Mean running time	824.6333118

Illustration 24: posix original



no of run time	Taken time(ns)	Taken time(s)
1	11354587285	113.5458729
2	11298409359	112.9840936
3	11776326462	117.7632646
5	11568225907	115.6822591
5	11287910643	112.8791064
6	11464536317	114.6453632
7	11370834474	113.7083447
8	11458387991	114.5838799
9	11469822224	114.6982222
10	11567825324	115.6782532
Mean running time	11461686599	114.616866

Illustration 25: Mean time Posix Thread



no of run time	Taken time(s)	Taken time(ns)
1	0.245558599	245558599
2	0.111947797	111947797
3	0.107625762	107625762
5	0.1116431	1116431
5	0.106913964	106913964
6	0.104076109	104076109
7	0.102828439	102828439
8	0.102628719	102628719
9	0.102856759	102856759
10	0.103060142	103060142
Mean running time	0.119913939	108861272.1

Illustration 26: Mean running time with Cuda

The Password Cracking runs faster from CUDA since CUDA program uses 26*26 CUDA core, which makes it faster than the other program.

3.1.1 CUDA Password cracking Code Analysis:

device: This function called and executed on devices.
Is_a_match(char *attempt): This function has four attempt variables and plain passwords array and also matches the password with attempts by running in while loop
global: It declares kernel which is called on host and executed on devices.
blockIdk: blockidk is similar to thread index while it refers to the number associated with the block.
ThreadIdk: Its number associated with each thread within the block.
Karnel<<<26,26>>>(): This function is invoked and will execute on device but is invoked from the host. The next part indicates that we want to execute the karnel with 26 thread block consisting of 26 thread.
CudaThreadSynchronized(): It returns an error if one of the preceding tasks has failed.

3.2 Image processing:

a) Create another version of the edge detection program that utilises CUDA. Each CUDA thread should process an individual pixel, therefore given an image that is 100 x 72 this will require 7200 threads.

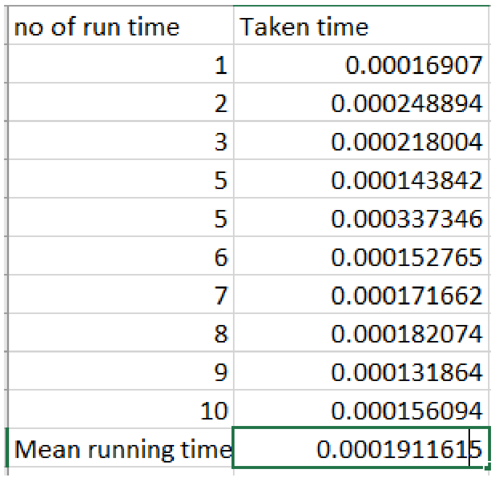
Answer:

Source code for Image processing with CUDA:

```
#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>
/************************************************************************
  Compile with:
    nvcc -o imagedetection_cuda imagedetection_cuda.cu -lglut -lGL

  Run With:
  ./imagedetection_cuda > results.txt

************/
#define width 100
#define height 72

unsigned char results[width * height];

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

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  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,
  255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,
  0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,255,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,0,255,255,0,0,255,255,255,
  255,255,255,0,255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,
  255,255,255,255,0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  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,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  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,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,0,255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,255,
  0,0,255,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,0,
  0,0,0,255,255,0,0,0,255,0,0,0,0,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,0,0,0,0,0,255,255,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,255,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,0,0,0,0,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,255,255,255,0,0,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,0,0,0,255,255,255,0,255,255,0,
  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,
  0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,0,0,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0
};

 __global__ void detect_edges(unsigned char *in, unsigned char *out) {
  int i = (blockIdx.x * 72) + threadIdx.x;
  int x, y; // the pixel of interest
  int b, d, f, h; // the pixels adjacent to x,y used for the calculation
  int r; // the result of calculate

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

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

      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:
      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) {

  unsigned char *d_results;
  unsigned char *d_image;

  cudaMalloc((void**)&d_image, sizeof(unsigned char) * (width * height));
  cudaMalloc((void**)&d_results, sizeof(unsigned char) * (width *
height));

  cudaMemcpy(d_image, &image, sizeof(unsigned char) * (width * height),
cudaMemcpyHostToDevice);
   signal(SIGINT, sigint_callback);


  struct timespec start, finish;
  long long int time_elapsed;

  clock_gettime(CLOCK_MONOTONIC, &start);
  detect_edges<<<100,72>>>(d_image, d_results);
  cudaThreadSynchronize();

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



  clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));
 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;
}
```
Illustration 27: Image processing program with Cuda

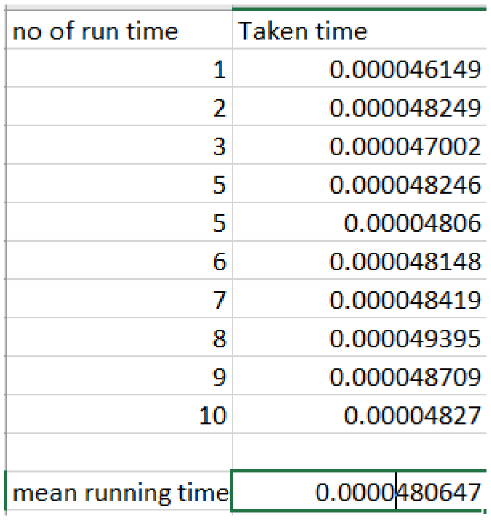
b) Compare the mean running time of the CUDA version with the original program and multithreaded versions. Comment of the effect of invoking more threads than cores.

Answer:



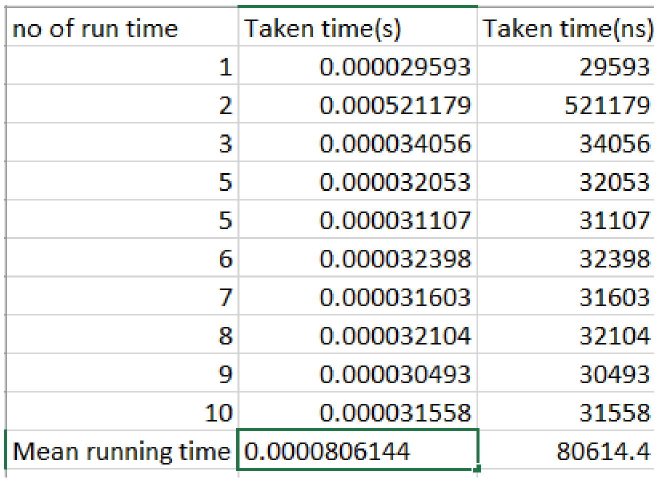
no of run time	Taken time
1	0.00016907
2	0.000248894
3	0.000218004
5	0.000143842
5	0.000337346
6	0.000152765
7	0.000171662
8	0.000182074
9	0.000131864
10	0.000156094
Mean running time	0.0001911615

Illustration 28: image processing original program



no of run time	Taken time
1	0.000046149
2	0.000048249
3	0.000047002
5	0.000048246
5	0.00004806
6	0.000048148
7	0.000048419
8	0.000049395
9	0.000048709
10	0.00004827
mean running time	0.0000480647

Illustration 29: image processing Posix Thread



no of run time	Taken time(s)	Taken time(ns)
1	0.000029593	29593
2	0.000521179	521179
3	0.000034056	34056
5	0.000032053	32053
5	0.000031107	31107
6	0.000032398	32398
7	0.000031603	31603
8	0.000032104	32104
9	0.000030493	30493
10	0.000031558	31558
Mean running time	0.0000806144	80614.4

Illustration 30: image processing with cuda

3.2.1 CUDA Image Processing Code Analysis:

Tidy_and_exit function called from the places in the program where the program terminates. It returns back to system memory that was allocated for pixels and also done in order to prevent memory leak.

Sigint_callback is called when user enters CLrt+C in keyword. It invoked tidy_and_exit so the program is terminated clearly.

In display function first bufferes is cleared then pixels is drawn and giFlush makes image appear in the screen.

Key_pressed(): This function is called when the users enters a key in keyword then the program is terminated if escape button is pressed.

cudaMalloc: allocates the memory pointer on GPU and its returns the address of the allocated memory block which is stored in variable with pointer.

CudaMemcy: Copy and image from the host to the devices

Detect_edges<<<100,72>>>(d_image,d_results): invokes the karnel function and karnel function to be run on GPU instead of CPU.

Copy and results from device to host:

CUDAMemcpy(&results, d_results, sizeof(Unsigned char)*(width*height), CudaMemcpyDevicesToHost;

OpenGl api provides large set of functions with which images and graphics can be manipulated or rendered. Some Functions are:

```
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();

3.3 Linear Regression:

a) Create another version of the linear regression program that utilises CUDA. There should be 1000 CUDA threads that can evaluate a single data point in parallel. The main algorithm therefore needs to call the CUDA kernel 8 times for each minimisation iteration.

Answer:

Source code for Linear regression program with cuda:

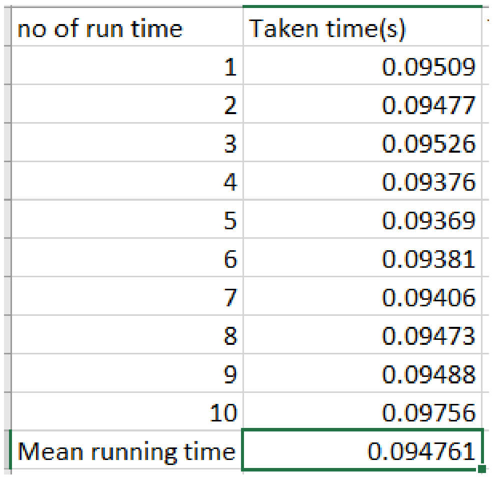
```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include <cuda_runtime_api.h>
#include <errno.h>
#include <unistd.h>

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

  To compile:
    nvcc -o linearregression_cuda linearregression_cuda.cu -lm

  To run:
    ./linearregression_cuda

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

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

int n_data = 1000;
__device__ int d_n_data = 1000;
```

```
point_t data[] = {
 {65.11,126.40},{76.79,149.00},{76.93,162.00},{65.24,113.46},
  {78.54,145.93},{84.60,161.77},{85.60,162.58},{82.32,152.21},
  {78.17,144.80},{69.47,142.78},{82.72,156.05},{11.56,52.20},
  {66.15,122.01},{75.13,145.75},{ 8.11,36.01},{71.58,150.44},
  {23.30,70.06},{42.59,86.42},{39.11,76.86},{ 8.77,36.29},
  {83.41,152.32},{ 3.44,36.86},{72.15,126.11},{66.29,129.15},
  {28.93,92.24},{91.62,172.01},{ 0.39,40.02},{55.24,104.88},
  {44.96,90.98},{89.66,170.29},{29.39,86.66},{56.19,109.96},
  {79.43,153.15},{54.27,110.73},{ 9.28,54.91},{31.16,76.74},
  {20.00,49.29},{67.25,122.68},{95.64,182.45},{66.03,128.30},
  {36.60,94.60},{83.93,120.16},{22.67,76.02},{81.17,164.59},
  {84.70,147.15},{34.58,87.81},{ 0.26,39.25},{82.07,149.43},
  { 2.63,41.39},{ 1.74, 7.95},{70.98,133.50},{16.65,49.48},
  {27.85,61.85},{55.84,105.63},{81.77,153.60},{19.81,61.26},
  {28.19,97.35},{ 2.62,32.52},{60.42,123.43},{53.67,118.83},
  {92.67,163.43},{ 4.09,30.06},{31.35,78.55},{54.79,103.97},
  {89.15,163.38},{20.35,66.02},{28.55,88.62},{11.66,58.29},
  {89.90,154.40},{ 0.14,51.92},{ 4.75,37.69},{53.83,108.99},
  {62.17,127.68},{79.10,133.64},{24.19,68.78},{51.41,100.86},
  {44.52,92.93},{23.02,66.51},{98.60,181.12},{ 6.05,48.82},
  {62.79,147.70},{ 5.06,50.58},{85.40,155.28},{12.33,60.17},
  {49.62,118.33},{ 9.03,48.29},{45.21,88.73},{28.22,55.37},
  {91.32,165.67},{ 6.74,44.19},{46.03,93.83},{69.75,139.69},
  { 2.15,40.31},{95.82,160.20},{ 6.64,54.91},{75.25,148.74},
  {39.64,68.97},{ 5.55,66.26},{90.53,155.37},{39.95,91.42},
  {68.89,132.98},{33.52,78.37},{15.84,38.51},{72.73,139.50},
  {21.54,73.78},{ 4.64,47.34},{66.57,132.87},{27.38,71.86},
  {93.83,181.33},{75.83,161.75},{26.47,56.70},{84.23,151.43},
  { 0.43,43.29},{88.50,160.27},{66.15,129.59},{78.31,141.68},
  {36.90,101.61},{71.78,139.52},{90.37,173.79},{ 0.58,45.98},
  {67.63,131.85},{57.43,100.37},{88.43,161.15},{74.83,132.98},
  {29.31,54.66},{79.06,146.78},{54.41,120.80},{51.76,108.96},
  {11.80,65.51},{38.19,90.48},{18.40,71.77},{76.29,148.07},
  {75.30,135.15},{59.56,126.34},{32.71,86.25},{42.35,116.15},
  { 4.85,38.50},{ 3.14,50.60},{48.27,90.59},{34.96,88.02},
  {10.03,50.01},{ 5.51,40.83},{68.32,136.16},{74.87,134.02},
  { 1.56,47.50},{19.52,72.68},{ 9.10,52.12},{50.79,102.10},
  {53.11,105.38},{94.93,174.68},{16.03,44.26},{13.26,49.58},
  { 3.24,46.86},{77.38,158.65},{21.57,62.81},{41.63,89.86},
  {13.55,59.72},{25.43,71.35},{86.73,166.79},{77.15,149.52},
  {26.47,64.94},{48.65,92.62},{33.66,75.10},{25.20,63.45},
  {25.46,86.18},{70.52,147.39},{98.32,175.47},{23.09,62.68},
  {48.90,118.74},{69.07,141.45},{50.54,132.25},{55.80,119.88},
  {25.65,92.01},{54.39,112.81},{79.86,165.28},{95.98,154.86},
  {48.14,108.06},{36.33,88.43},{ 6.34,35.96},{86.04,151.77},
  {57.03,116.32},{97.95,180.12},{29.66,73.83},{12.52,35.04},
  {43.93,83.56},{33.63,78.69},{64.00,128.13},{14.49,50.14},
  {49.66,112.89},{82.54,162.20},{81.92,143.19},{28.07,78.90},
  {14.26,47.92},{23.97,63.31},{27.69,73.01},{78.13,119.54},
  {34.43,82.48},{66.13,123.89},{61.84,135.81},{17.03,57.30},
  { 5.61,52.51},{34.44,88.49},{17.81,81.52},{34.26,79.71},
  {93.17,161.29},{ 8.10,39.44},{93.51,158.23},{61.48,133.51},
  {27.22,71.93},{17.11,50.22},{27.73,81.68},{16.07,61.27},
  {63.81,122.63},{ 0.27,36.83},{62.21,120.74},{42.36,85.01},
  {60.61,143.23},{68.59,121.10},{28.48,68.27},{23.39,71.50},
  {93.40,162.60},{50.72,114.87},{24.53,80.83},{92.00,160.38},
  {79.29,175.12},{28.84,78.42},{13.79,44.14},{23.18,62.24},
  {69.07,122.23},{41.93,120.63},{62.32,125.07},{72.39,136.08},
  {41.86,92.41},{ 2.35,21.43},{56.14,133.02},{33.91,90.07},
  {13.68,76.01},{14.55,71.51},{73.79,152.07},{33.47,97.28},
  {31.12,65.68},{ 4.33,41.19},{22.94,58.10},{85.12,160.37},
  {80.26,154.39},{37.01,78.54},{ 6.94,38.10},{ 7.83,60.51},
  {42.44,90.12},{26.69,91.63},{99.36,184.47},{ 9.33,55.05},
  {16.87,63.97},{32.41,80.49},{36.34,77.15},{59.91,122.62},
  { 7.35,34.97},{99.21,183.69},{34.07,97.06},{43.85,102.96},
  {55.20,111.99},{ 3.95,41.47},{26.71,82.57},{16.69,64.61},
  {38.32,96.85},{76.67,136.21},{86.22,175.61},{35.18,71.39},
  {57.39,117.85},{72.12,139.30},{90.19,173.32},{97.26,163.25},
  {82.08,135.04},{40.69,96.78},{25.49,75.76},{83.38,149.39},
  {63.64,135.54},{90.52,166.25},{79.96,154.68},{45.70,107.42},
  {15.54,61.07},{97.10,170.63},{41.10,87.33},{35.86,74.41},
  {57.22,120.65},{16.28,64.80},{46.33,97.53},{31.84,83.82},
  {90.15,177.85},{13.39,77.75},{ 8.25,26.83},{91.74,155.44},
  {11.65,61.09},{26.30,82.75},{61.72,128.34},{76.94,152.59},
  {26.70,81.25},{ 6.11,39.57},{97.46,172.22},{13.50,43.37},
  {16.46,54.81},{44.36,84.30},{45.83,105.17},{41.47,105.60},
  {31.72,82.66},{58.79,113.04},{95.35,168.80},{27.91,73.77},
  {61.28,126.49},{ 1.18,32.24},{ 4.17,41.67},{79.08,142.98},
  { 4.80,37.58},{94.98,160.79},{37.47,80.15},{ 6.82,53.58},
  {57.54,118.51},{73.31,139.05},{91.40,166.67},{98.07,162.54},
  { 3.87,41.53},{63.71,130.41},{75.78,137.34},{56.32,122.22},
  { 8.03,63.52},{22.60,60.09},{94.56,158.12},{16.10,72.60},
  {82.22,155.28},{57.63,114.94},{55.26,118.38},{73.52,126.54},
  {59.13,123.52},{81.11,167.56},{68.73,123.16},{43.78,122.00},
  {27.48,70.82},{43.92,110.09},{ 7.04,44.70},{91.03,147.13},
  {44.55,114.71},{68.40,122.77},{ 6.31,55.15},{12.03,51.80},
  {26.62,77.09},{12.90,60.85},{41.47,115.00},{75.98,156.05},
  {62.84,118.68},{ 3.19,54.74},{74.93,132.72},{89.37,170.57},
  {57.12,114.25},{63.07,104.90},{60.20,129.41},{36.04,82.56},
  {66.43,133.16},{ 7.01,45.98},{87.68,162.25},{36.24,86.35},
  {60.38,140.39},{ 3.56,41.80},{65.74,138.56},{16.34,61.32},
  {34.00,105.97},{77.42,132.46},{61.79,135.32},{61.13,116.51},
  {90.22,159.21},{68.78,137.53},{48.35,110.41},{58.48,110.06},
  { 0.04,45.54},{87.14,152.92},{73.42,154.90},{48.73,105.35},
  {36.26,97.55},{50.34,107.46},{95.65,162.87},{11.76,50.19},
  {12.83,58.44},{ 7.59,65.35},{51.44,109.31},{15.39,65.30},
  {83.69,147.00},{75.86,139.91},{25.87,87.89},{ 0.79,40.68},
  { 4.87,62.14},{64.71,126.73},{60.94,111.46},{82.18,142.90},
  {21.67,55.08},{33.20,93.76},{11.93,64.07},{10.59,39.64},
  {20.90,63.85},{21.47,81.20},{15.02,80.95},{67.04,112.88},
  { 0.78,34.00},{24.49,77.44},{ 0.49,33.31},{70.88,138.01},
  {33.07,81.36},{74.11,139.95},{ 1.26,40.24},{26.68,81.03},
  {81.37,155.39},{89.28,163.35},{82.92,150.44},{88.94,166.64},
  {14.35,63.72},{58.92,119.69},{47.88,100.06},{73.51,145.81},
  {21.11,79.13},{33.98,87.52},{87.88,158.32},{24.47,79.36},
  {34.00,92.04},{42.25,105.99},{20.75,64.71},{65.45,131.84},
  {10.51,38.23},{36.26,87.53},{70.72,129.61},{62.87,129.51},
  {88.93,170.73},{50.72,124.30},{52.28,104.85},{ 1.38,36.72},
  {93.31,171.99},{56.94,135.14},{99.80,174.15},{ 2.74,46.10},
  {42.39,81.82},{42.07,102.65},{ 3.06,34.94},{69.46,129.37},
  {43.33,104.41},{69.47,129.06},{68.70,134.00},{95.28,179.59},
  {66.66,132.99},{14.40,70.02},{39.22,77.48},{ 5.87,32.85},
  {71.70,144.19},{69.40,140.87},{51.33,113.42},{38.58,97.94},
  {70.18,129.30},{28.53,76.70},{11.77,52.53},{16.26,69.96},
  {86.39,161.92},{14.35,60.74},{17.32,53.69},{60.10,117.59},
  {55.86,117.63},{ 9.26,58.48},{48.28,100.74},{79.81,149.90},
  {59.38,105.17},{72.31,119.55},{41.23,91.88},{70.20,136.75},
  {82.58,158.00},{72.29,130.76},{10.80,60.44},{81.60,167.47},
  {57.21,120.91},{83.97,150.98},{78.97,152.79},{78.71,146.61},
  {98.28,172.82},{39.89,87.34},{92.46,169.18},{29.94,69.32},
  {64.78,127.52},{52.32,113.01},{29.54,78.81},{46.04,97.77},
  {97.71,155.81},{90.08,172.72},{59.58,117.24},{74.61,149.80},
  {64.20,116.45},{42.14,102.85},{44.27,107.95},{97.48,174.75},
  {84.52,164.71},{19.46,51.01},{87.05,166.73},{28.47,61.63},
  { 0.59,40.82},{49.83,100.11},{25.05,62.68},{20.32,62.69},
  {56.64,126.50},{15.41,59.79},{98.36,173.55},{57.73,97.55},
  {29.88,86.10},{26.44,65.10},{92.31,174.76},{10.74,49.39},
  {88.32,163.82},{71.67,156.47},{94.40,181.01},{16.04,55.25},
  {76.78,141.43},{81.76,146.31},{81.41,144.40},{ 7.59,65.42},
  {29.83,93.19},{71.85,128.96},{17.45,45.91},{66.78,148.29},
  {87.12,149.00},{55.37,115.72},{ 8.34,69.56},{10.30,58.08},
  {26.49,79.04},{33.65,83.88},{82.84,146.92},{74.19,132.28},
  {47.38,105.49},{95.30,172.52},{ 8.60,54.66},{38.14,69.96},
  {14.06,65.00},{27.26,74.14},{31.28,82.60},{24.83,97.76},
  {53.38,101.53},{26.88,87.40},{77.79,153.42},{ 4.60,53.07},
  {36.70,76.53},{33.96,93.76},{64.04,116.47},{73.85,156.81},
  {31.88,85.18},{10.44,62.68},{41.55,99.48},{31.20,94.01},
  {69.63,137.46},{30.90,92.46},{54.24,103.71},{82.12,149.86},
  {57.43,119.86},{16.83,61.60},{38.45,74.99},{ 9.38,60.12},
  {18.91,53.31},{65.28,115.98},{52.45,119.66},{ 4.88,52.30},
  {49.92,95.81},{60.44,126.35},{25.07,83.25},{58.21,108.57},
  {28.81,94.01},{44.94,99.55},{35.75,79.36},{95.26,164.51},
  {35.14,113.38},{ 6.95,51.77},{37.56,81.49},{27.79,72.55},
  {68.04,129.74},{19.46,82.90},{13.49,40.02},{40.81,92.01},
  {44.13,86.46},{90.16,178.62},{82.34,153.55},{92.32,165.64},
  {78.20,166.78},{24.76,66.00},{91.00,175.56},{85.94,172.81},
  {98.74,178.91},{ 7.47,34.02},{37.28,85.95},{ 8.94,54.67},
  {31.78,85.85},{31.71,82.87},{44.29,96.83},{21.85,68.96},
  {15.20,48.69},{ 9.51,58.55},{14.53,53.46},{87.25,166.09},
  {35.25,79.77},{45.43,106.79},{16.24,69.34},{61.36,142.83},
  {99.33,176.18},{ 2.66,42.92},{42.69,105.85},{69.04,124.32},
  {62.77,125.38},{87.76,149.94},{68.38,124.70},{44.95,109.62},
  { 8.36,63.51},{29.47,75.02},{42.49,87.92},{29.05,95.14},
  { 1.36,43.70},{60.36,102.16},{23.57,54.88},{30.84,80.74},
  {10.19,42.03},{97.59,177.44},{36.08,89.31},{21.74,45.86},
  {58.56,113.61},{34.10,92.54},{87.76,174.79},{43.42,107.45},
  {55.01,110.06},{45.87,119.35},{21.24,61.64},{ 0.63,23.13},
  {44.94,99.54},{ 5.22,47.01},{ 1.71,42.19},{92.32,159.09},
  {28.15,76.89},{77.98,128.92},{40.11,84.47},{80.44,144.10},
  {21.62,80.78},{27.18,70.12},{80.83,148.92},{65.54,132.52},
  {69.13,124.43},{26.54,59.95},{ 0.13,36.97},{24.07,70.64},
  {27.58,70.42},{45.07,121.14},{11.82,46.41},{81.39,156.60},
  {49.46,95.96},{56.25,93.87},{92.93,167.21},{85.35,169.34},
  {32.46,93.55},{37.88,93.61},{66.98,144.61},{67.21,133.07},
  {37.90,81.47},{68.35,136.90},{69.28,140.78},{78.26,143.36},
  {28.73,69.02},{48.85,91.09},{ 6.11,61.37},{69.24,156.76},
  {58.32,123.43},{13.23,59.97},{32.85,74.58},{48.15,120.84},
  {74.60,145.21},{46.64,104.61},{63.37,120.76},{13.36,59.46},
  {69.89,142.17},{67.89,136.10},{49.22,99.19},{73.16,143.29},
  {47.79,130.64},{41.71,94.63},{93.46,171.77},{99.74,185.80},
  {58.15,112.90},{24.90,82.06},{17.53,58.51},{34.06,80.58},
  {51.11,115.72},{19.12,64.68},{29.05,80.50},{30.71,91.87},
  {20.00,77.43},{38.82,97.86},{25.56,71.28},{24.69,51.78},
  {15.15,52.31},{89.92,178.26},{97.21,171.26},{54.16,134.74},
  {84.67,149.81},{74.93,123.09},{ 5.26,24.90},{99.04,183.04},
  {89.69,180.72},{ 9.57,59.78},{27.52,86.77},{ 7.79,63.47},
  {86.70,159.99},{12.54,49.44},{65.80,139.16},{60.68,99.45},
  {37.01,99.10},{65.32,128.72},{79.27,139.94},{13.48,59.51},
  {16.15,65.81},{ 5.50,56.27},{21.44,61.06},{17.95,80.39},
  {22.99,69.66},{78.04,139.81},{ 8.19,45.53},{53.04,114.50},
  {22.03,55.53},{71.11,134.99},{12.41,60.57},{47.53,107.37},
  { 0.20,27.63},{ 3.31,26.26},{59.81,132.51},{50.17,104.10},
  {84.25,141.14},{91.89,171.66},{14.95,62.25},{ 9.00,61.06},
  {29.68,75.94},{98.55,169.15},{63.95,127.72},{41.36,82.03},
  {92.20,168.84},{71.55,142.74},{89.17,168.56},{36.19,84.82},
  { 5.83,58.93},{32.71,82.95},{13.63,72.12},{20.78,69.59},
  {96.66,156.89},{40.74,93.92},{12.50,64.55},{91.70,165.65},
  {45.68,89.74},{10.70,52.42},{80.60,159.09},{46.91,99.34},
  {42.30,97.16},{34.03,85.62},{68.84,132.20},{94.47,166.73},
  { 6.57,23.33},{88.09,172.72},{10.29,44.01},{16.28,64.39},
  {40.21,82.53},{42.50,101.48},{85.18,145.73},{88.49,176.79},
  {23.93,69.17},{21.58,71.42},{43.56,101.34},{18.85,72.03},
  { 4.01,20.86},{58.74,130.89},{ 0.55,42.23},{64.01,138.48},
  {86.32,164.34},{ 4.01,62.96},{71.65,145.59},{59.98,128.80},
  {47.29,107.25},{52.80,112.62},{73.48,143.42},{60.71,105.76},
  {14.39,46.36},{91.65,166.65},{68.70,134.37},{17.20,63.05},
  {49.86,111.33},{15.66,66.77},{13.85,55.13},{11.74,62.94},
  {46.11,92.86},{90.43,144.43},{12.80,46.53},{ 8.49,48.78},
  {92.34,176.52},{77.18,145.53},{18.95,72.13},{25.16,77.45},
  {79.17,156.72},{94.54,168.51},{12.56,52.73},{31.32,80.71},
  {83.67,145.98},{69.02,141.65},{16.67,51.28},{43.22,108.69},
  { 2.77,44.28},{28.19,92.70},{85.57,161.86},{16.23,62.41},
  { 7.59,70.56},{36.61,85.26},{31.17,83.60},{77.49,151.10},
  {12.82,38.79},{30.11,81.59},{50.07,122.10},{74.50,144.63},
  {94.48,175.21},{82.49,146.39},{47.18,90.69},{19.81,68.22},
  {67.87,135.07},{86.53,158.63},{ 4.02,67.70},{79.22,163.68},
  {18.63,65.68},{93.39,170.96},{95.97,163.34},{75.47,121.35},
  { 0.78,37.11},{ 9.53,50.40},{39.13,110.03},{95.69,168.67},
  {27.61,84.96},{47.10,120.52},{96.66,178.29},{88.15,179.79},
  {54.08,127.28},{98.67,173.36},{28.33,79.71},{ 3.98,31.32},
  {98.84,179.12},{22.71,70.55},{ 2.25,35.21},{32.51,72.10},
  {61.33,121.66},{70.04,137.97},{47.57,129.83},{15.27,63.70},
  {67.47,148.68},{90.29,162.66},{ 5.58,56.85},{26.24,75.05},
  {97.20,190.42},{97.93,174.98},{72.40,139.24},{36.57,100.59},
  { 9.55,69.48},{28.55,80.48},{23.97,69.20},{40.40,94.65},
  {93.43,169.59},{56.99,101.50},{29.82,78.34},{63.85,105.18},
  {36.57,93.67},{29.99,100.46},{48.09,99.60},{17.13,85.66},
  {42.67,102.26},{26.34,76.52},{ 9.81,48.84},{35.70,76.14},
  {89.40,153.75},{97.80,177.01},{27.89,69.25},{46.43,113.97},
  {21.64,62.84},{72.79,131.04},{86.23,150.89},{57.53,122.30},
  {36.87,91.32},{13.15,50.63},{81.13,165.31},{29.36,108.50},
  {25.65,81.54},{21.91,58.02},{60.06,128.34},{53.90,116.97},
  {37.20,91.22},{ 2.75,54.02},{78.84,143.43},{78.42,129.08},
  {30.30,91.45},{ 6.19,51.64},{15.94,75.49},{49.50,107.77},
  {48.58,103.97},{42.05,114.46},{98.55,169.49},{23.59,77.96},
  { 4.95,31.04},{51.61,122.22},{89.57,166.43},{97.29,183.00},
  {67.36,143.50},{70.70,143.79},{ 7.09,61.57},{ 4.55,35.73},
  { 3.12,26.08},{27.61,71.71},{17.87,65.37},{73.82,148.35},
  {71.86,152.17},{39.75,97.64},{11.52,51.38},{84.82,150.22},
  {33.13,77.12},{34.83,83.95},{53.84,105.93},{85.86,161.20},
  {80.36,135.81},{29.48,66.91},{33.44,84.75},{27.94,89.60},
  {61.89,130.52},{15.65,50.50},{66.84,126.11},{61.89,124.02},
  {30.64,82.56},{63.67,113.67},{93.79,175.50},{89.78,180.21},
  {49.60,106.06},{78.60,152.09},{88.82,171.67},{ 4.49,41.76},
  {12.41,62.47},{57.54,122.63},{42.00,96.54},{15.89,62.16},
  {18.09,43.62},{98.19,177.35},{49.84,105.13},{59.38,128.63},
  {55.34,118.20},{60.21,125.47},{31.34,69.51},{79.20,139.77},
  {26.37,81.64},{45.32,72.27},{91.13,173.22},{91.36,169.43},
  {65.10,128.76},{24.33,59.90},{39.37,93.39},{88.88,156.65},
  {66.86,146.50},{73.40,126.02},{14.09,64.93},{87.34,173.83},
  {18.26,68.89},{92.26,160.92},{77.91,157.56},{52.89,98.98},
  {38.14,109.31},{41.50,96.53},{26.81,89.59},{47.42,103.41},
  {68.58,132.42},{60.29,126.09},{64.99,125.45},{76.35,144.33},
  {11.69,57.61},{28.16,72.44},{23.94,72.18},{95.67,182.61},
  {59.17,118.32},{35.19,83.30},{19.53,74.53},{45.16,96.72},
  {66.63,128.79},{96.13,182.09},{65.31,126.98},{33.27,102.77},
  { 3.65,39.52},{19.26,74.36},{32.61,70.26},{37.77,82.99},
  { 1.77,32.37},{87.50,167.27},{90.60,158.93},{86.81,154.12},
  {23.83,77.55},{97.47,166.55},{83.99,167.80},{44.51,104.49},
  {86.46,168.85},{75.17,142.50},{83.71,173.31},{92.83,162.93}
};

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);

        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,
         cudaGetErrorString(error));
       exit(1);
       }


       error = cudaMalloc(&d_data, sizeof(data));
       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);
        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];
        best_error_i = i;
      }

                      error_sum_total = 0;
    }


    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));
              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\n", bm, bc, be);

       clock_gettime(CLOCK_MONOTONIC, &finish);

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

  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));

  return 0;
}
```
Illustration 31: Linear regression with cuda

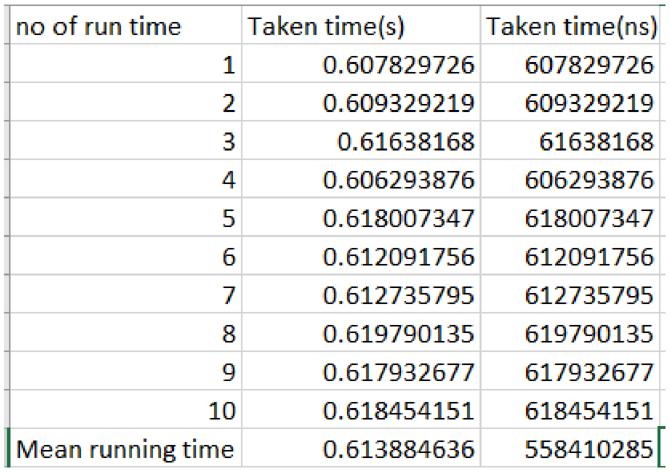
b) Compare the mean running time of the CUDA version with the original program and multithreaded versions.

Answer:



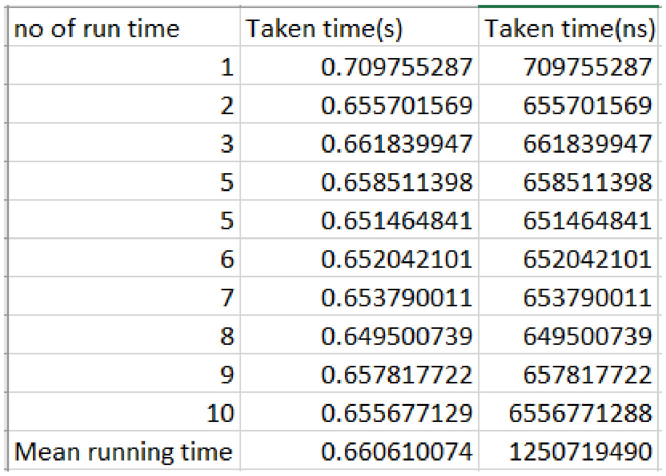
no of run time	Taken time(s)
1	0.09509
2	0.09477
3	0.09526
4	0.09376
5	0.09369
6	0.09381
7	0.09406
8	0.09473
9	0.09488
10	0.09756
Mean running time	0.094761

Illustration 33: linear Regression original program



no of run time	Taken time(s)	Taken time(ns)
1	0.607829726	607829726
2	0.609329219	609329219
3	0.61638168	61638168
4	0.606293876	606293876
5	0.618007347	618007347
6	0.612091756	612091756
7	0.612735795	612735795
8	0.619790135	619790135
9	0.617932677	617932677
10	0.618454151	618454151
Mean running time	0.613884636	558410285

Illustration 33:linear Regression with posix thread



no of run time	Taken time(s)	Taken time(ns)
1	0.709755287	709755287
2	0.655701569	655701569
3	0.661839947	661839947
5	0.658511398	658511398
5	0.651464841	651464841
6	0.652042101	652042101
7	0.653790011	653790011
8	0.649500739	649500739
9	0.657817722	657817722
10	0.655677129	6556771288
Mean running time	0.660610074	1250719490

Illustration 33: linear Regression with cuda

3.3.1 CUDA Linear Regression Code Analysis:

The main object of this code is to find out the minimum m and c value with best error using CUDA tooklit. M and c are the slope of line and intercept of the staright line. Slope deals with direction and lines steepness where intercepts are points where it crosses the axis. There are intercepts x-intercept and y- inetcept. We tried to findout the values of m and c with optimal root mean square error.

We have to declared the bm, bc and best_error with 1.3, 10 and 99999999. We have also declared the best error in the program. Two time-spec start and finish and finish along with elapsed is declared in declaration phase to access the start time , finish time and the time difference. Clock_get time function is used to get the time from the system which takes two parameters CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time.

4. MPI:

4.1 Password Cracking:

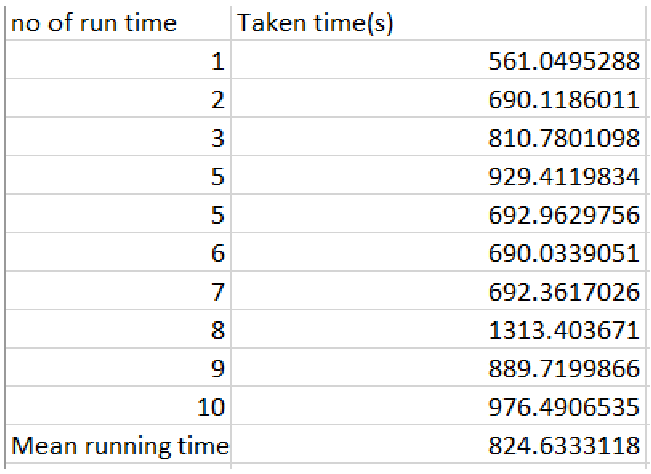
a) Create another version of the two-initials-four-digits password cracker. A master should share the work out to two compute instances. i.e. instance 1 should explore AA through to MZ, whilst instance 2 should explore NA through to NZ.

```
/*********************************************************************
****
  Password Cracking using MPI

  To compile:
     mpicc -o passwordcrackingwith_mpi passwordcrackingwith_mpi.c -lrt -
lcrypt

  To run 3 processes on this computer:
    mpirun -n 3 ./passwordcrackingwith_mpi

*********************************************************************
***/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <crypt.h>
#include <mpi.h>
```

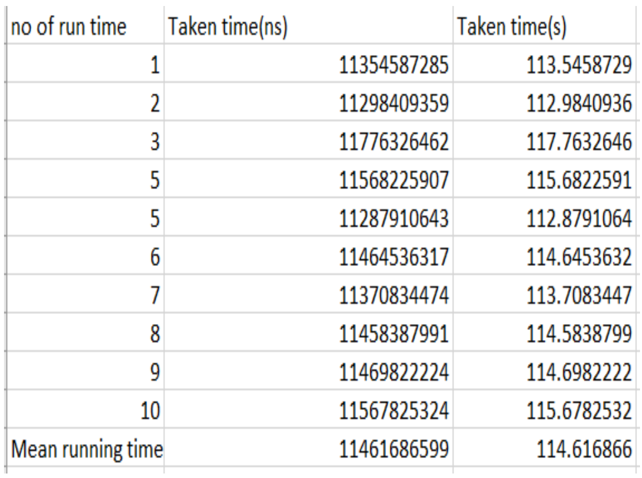
```c
int n_passwords = 4;
pthread_t t1, t2;
char *encrypted_passwords[] = {

"$6$KB$0G24VuNaA9ApVG4z8LkI/OOr9a54nBfzgQjbebhqBZxMHNg0HiYYf1Lx/HcGg6q1nnO
SArPtZYbGy7yc5V.wP/",

"$6$KB$WksuNcTfYjZWjDC4Zt3ZAmQ38OrsWwHyDgf/grFJ2Sgg.qpOz56lMpBVfWYdQZa9Pks
a2TJRVYVb3K.mbYx4Y1",

"$6$KB$UdJ/FGlqWHrXeWFVdjwqMel4WRTW93ai6K891ug/Td3NnAWj1AMMfZkQGut4Ia7hpWb
4ECic6xlvF.BGJdOj90",

"$6$KB$mV33QckPvVM55rLtO3QTXr5ib3rvmyndjSWLt0DZSOimZ0bM/djcZRyTY0fm25xKc/u
5b.aTNjV8mBxv9ESTL0"
};

void substr(char *dest, char *src, int start, int length){
  memcpy(dest, src + start, length);
  *(dest + length) = '\0';
}

void kernel_function1(char *salt_and_encrypted){
  int x, y, z;     // Loop counters
  char salt[7];    // String used in hashing the password. Need space

  char plain[7];   // The combination of letters currently being checked
  char *enc;       // Pointer to the encrypted password
  int count = 0;   // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='A'; x<='M'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
       printf("Instance 1:");
       sprintf(plain, "%c%c%02d",x, y, z);
       enc = (char *) crypt(plain, salt);
       count++;
       if(strcmp(salt_and_encrypted, enc) == 0){
         printf("#%-8d%s %s\n", count, plain, enc);
       } else {
         printf(" %-8d%s %s\n", count, plain, enc);
       }
      }
    }
  }
  printf("%d solutions explored\n", count);
}
void kernel_function2(char *salt_and_encrypted){
  int x, y, z;     // Loop counters
  char salt[7];    // String used in hashing the password. Need space
  char plain[7];   // The combination of letters currently being checked
  char *enc;       // Pointer to the encrypted password
  int count = 0;   // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='N'; x<='Z'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
       printf("Instance 2:");
       sprintf(plain, "%c%c%02d",x, y, z);
       enc = (char *) crypt(plain, salt);
       count++;
       if(strcmp(salt_and_encrypted, enc) == 0){
         printf("#%-8d%s %s\n", count, plain, enc);
       } else {
         printf(" %-8d%s %s\n", count, plain, enc);
       }
      }
    }
  }
  printf("%d solutions explored\n", count);
}


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) {
 struct timespec start, finish;
  long long int time_elapsed;

  clock_gettime(CLOCK_MONOTONIC, &start);


  int size, rank;
int i;

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if(size != 3) {
    if(rank == 0) {
      printf("This program needs to run on exactly 3 processes\n");
    }
  } else {
    if(rank ==0){

      int x;

      MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
      MPI_Send(&x, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);



    } else if(rank==1){
      int number;
      MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
                         MPI_STATUS_IGNORE);
        for(i=0;i<n_passwords;i<i++) {
          kernel_function1(encrypted_passwords[i]);
        }
      }
      else{
      int number;
      MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
                         MPI_STATUS_IGNORE);
        for(i=0;i<n_passwords;i<i++) {
          kernel_function2(encrypted_passwords[i]);
        }
      }
    }
    MPI_Finalize();
 clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));

  return 0;
}
```
Illustration 34: Password cracking with MPI (2initial 2 digits)

b) Compare the mean running time of the MPI version with the original, multithread and CUDA versions.

Answer:

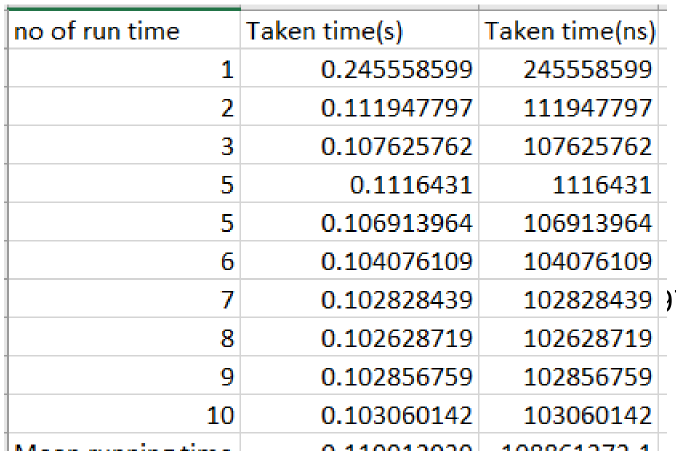


no of run time	Taken time(s)
1	561.0495288
2	690.1186011
3	810.7801098
5	929.4119834
5	692.9629756
6	690.0339051
7	692.3617026
8	1313.403671
9	889.7199866
10	976.4906535
Mean running time	824.6333118

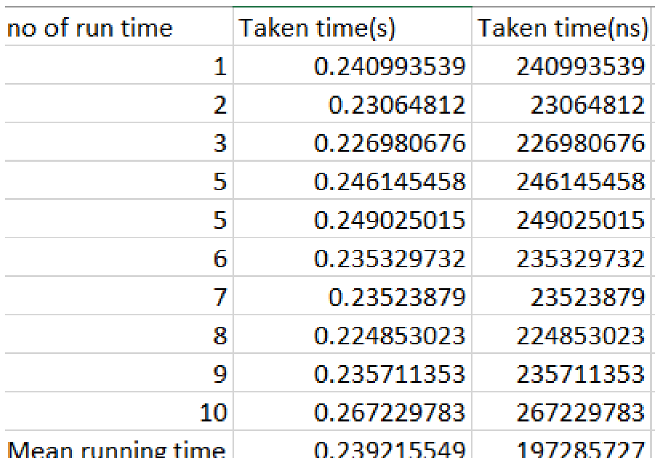


no of run time	Taken time(ns)	Taken time(s)
1	11354587285	113.5458729
2	11298409359	112.9840936
3	11776326462	117.7632646
5	11568225907	115.6822591
5	11287910643	112.8791064
6	11464536317	114.6453632
7	11370834474	113.7083447
8	11458387991	114.5838799
9	11469822224	114.6982222
10	11567825324	115.6782532
Mean running time	11461686599	114.616866

Illustration 35: password crack original Illustration 36: password crack with posix thread (posix)



no of run time	Taken time(s)	Taken time(ns)
1	0.245558599	245558599
2	0.111947797	111947797
3	0.107625762	107625762
5	0.1116431	1116431
5	0.106913964	106913964
6	0.104076109	104076109
7	0.102828439	102828439
8	0.102628719	102628719
9	0.102856759	102856759
10	0.103060142	103060142
Mean running time	0.110013030	108861273.1



no of run time	Taken time(s)	Taken time(ns)
1	0.240993539	240993539
2	0.23064812	23064812
3	0.226980676	226980676
5	0.246145458	246145458
5	0.249025015	249025015
6	0.235329732	235329732
7	0.23523879	23523879
8	0.224853023	224853023
9	0.235711353	235711353
10	0.267229783	267229783
Mean running time	0.239215549	197285727

Illustration 37: password crack with cuda   Illustration 38: password crack with mpi

The Password Cracking runs faster from CUDA since CUDA program uses 26*26 CUDA core which makes it faster than the other program. But looking below at the results MPI is comparatively faster than CUDA, sometimes slow processing of the device can also make a huge difference on the execution of the program.

Source code for a mpi 2initial 4 digits program:

```
/***************************************************************************
****
  Password Cracking using MPI

  To compile:
     mpicc -o 2initial4digitspasswordcrackwith_mpi
2initial4digitspasswordcrackwith_mpi.c -lrt -lcrypt

  To run 3 processes on this computer:
    mpirun -n 3 ./2initial4digitspasswordcrackwith_mpi


***************************************************************************
***/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <crypt.h>
#include <mpi.h>


int n_passwords = 4;
pthread_t t1, t2;
char *encrypted_passwords[] = {
"$6$KB$sl22ZaAfqOFudnKP6DKkTZOyZe1EaCbquYEzI2dgAE2/ngaDDWrP2t4qinJYOyABjxM
ddoPksNymjZz77xW0p/",
```

```c
"$6$KB$2u8TbEcS88ozs/AEDXxIE1d.YKoPWQUg/OksraIuZapCb.NLiH3xR.wEKmvnTXBtw1z
Gc1nmTDFlOGzT0fX0z1",

"$6$KB$6ege3SGHgG/LAXwhE6HpZYbTU9l29aU0ADFHnnfLNQqvvsPwtNoYnmFOI0VYm/3G8Gm
OgqLJ1x.H8S7AVAsme0",

"$6$KB$A37oXWARc8SH4/EZTSeN.MerLmjV4KJmdExETRQQraAhEORkTBqsWb4mSf4ZSxrndCS
w7I2wy4cOGOppiI4rc."
};

void substr(char *dest, char *src, int start, int length){
  memcpy(dest, src + start, length);
  *(dest + length) = '\0';
}

void kernel_function1(char *salt_and_encrypted){
  int x, y, z,a;       // Loop counters
  char salt[7];      // String used in hashing the password. Need space

  char plain[7];     // The combination of letters currently being checked
  char *enc;         // Pointer to the encrypted password
  int count = 0;     // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);

  for(x='A'; x<='M'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
       for(a=0; a<=99; z++){
       printf("Instance 1:");
       sprintf(plain, "%c%c%02d%02d",x, y, z,a);
       enc = (char *) crypt(plain, salt);
       count++;
       if(strcmp(salt_and_encrypted, enc) == 0){
         printf("#%-8d%s %s\n", count, plain, enc);
       } else {
         printf(" %-8d%s %s\n", count, plain, enc);
       }
}
      }
    }
  }
  printf("%d solutions explored\n", count);
}
void kernel_function2(char *salt_and_encrypted){
  int x, y, z,a;       // Loop counters
  char salt[7];      // String used in hashing the password. Need space

  char plain[7];     // The combination of letters currently being checked
  char *enc;         // Pointer to the encrypted password
  int count = 0;     // The number of combinations explored so far

  substr(salt, salt_and_encrypted, 0, 6);
  for(x='N'; x<='Z'; x++){
    for(y='A'; y<='Z'; y++){
      for(z=0; z<=99; z++){
  for(a=0; a<=99; z++){
        printf("Instance 2:");
        sprintf(plain, "%c%c%02d%02d",x, y, z,a);
        enc = (char *) crypt(plain, salt);
        count++;
        if(strcmp(salt_and_encrypted, enc) == 0){
          printf("#%-8d%s %s\n", count, plain, enc);
        } else {
          printf(" %-8d%s %s\n", count, plain, enc);
        }
}
      }
    }
  }
  printf("%d solutions explored\n", count);
}


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) {
 struct timespec start, finish;
  long long int time_elapsed;

  clock_gettime(CLOCK_MONOTONIC, &start);


  int size, rank;
int i;

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if(size != 3) {
    if(rank == 0) {
      printf("This program needs to run on exactly 3 processes\n");
    }
  } else {
    if(rank ==0){
      int x;

      MPI_Send(&x, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
      MPI_Send(&x, 1, MPI_INT, 2, 0, MPI_COMM_WORLD);



    } else if(rank==1){
      int number;
      MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
                         MPI_STATUS_IGNORE);
        for(i=0;i<n_passwords;i<i++) {
          kernel_function1(encrypted_passwords[i]);
        }
      }
      else{
      int number;
      MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
                         MPI_STATUS_IGNORE);
        for(i=0;i<n_passwords;i<i++) {
          kernel_function2(encrypted_passwords[i]);
        }
      }
    }
    MPI_Finalize();
 clock_gettime(CLOCK_MONOTONIC, &finish);
  time_difference(&start, &finish, &time_elapsed);
  printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));

  return 0;
}
```
Illustration 39: password crack with mpi(2initial 4 digits)

4.1.1 MPI Password Cracking Code Analysis:

mr.py: used to run the password cracking program 10 times in one go.

time_diff.c: used to capture the execution time of the program.

EncryptSHA512.c: used to encrypt 3 initials and 2 digits password or used to encrypt 2 initials and 4 digits password.

The main aim os this code is to find out the run time to crack the password using message passing interface. To calculate time two time-spec start and finish along with time elapsed is declared in declaration phase access the start time, finish time difference. Clock_get time function is used to get the time from the system which takes two parameters

CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time. MPI library is initialized by MPI_Init datatype. MPI_Comm_Size() function is used to identify size of communicator.MPI_Comm_rank() function is used to identify the rank of the calling process. the program only executed if there are exactly three process. The MPi program MPI_send () is used to do blocking send to other rank from current rank.

We have sent our work from rank 0 to rank1 and rank 2.so. from rank 1 and rank 2 we have called the function karnel_function1() and karnel_function2() after receiving the message from rank 0. Karnel_function1() tries to crack the password whose starting alphabets from A to M. and karnel_funtion2() tries to crack the password whose starting alphabets from N to Z. Three loop variables x ,y, z are used to find the matching password.

4.2 Image processing:

a) Create another version of the edge detection program that uses 4 MPI instances to each process a quarter of the image in a horizontal band. A master instance should share out the work and collate the results back into one image.

Answer:

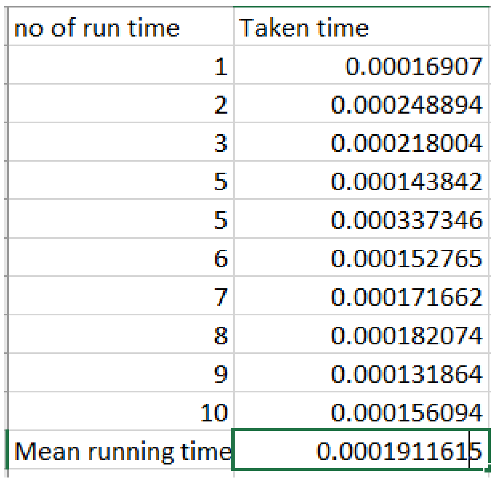
```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <GL/glut.h>
#include <GL/gl.h>
#include <malloc.h>
#include <signal.h>
#include <math.h>
/*************************************************************************

  To compile adapt the code below wo match your filenames:
    mpicc -o imageProcessingwith_mpi imageProcessingwith_mpi.c -lglut -lGL
-lm

  To run
    mpirun -n 5 -quiet ./imageProcessingwith_mpi
**************************************************************************
****/
#define width 100
#define height 72

int first, last;
unsigned char image[], results[width * height];

void detect_edges(unsigned char *in, unsigned char *out) {
  int i;
  int n_pixels = width * height;

  for(i=0;i<n_pixels;i++) {
    int x, y; // the pixel of interest
    int b, d, f, h; // the pixels adjacent to x,y used for the calculation
    int r; // the result of calculate

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

      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;

      case 'e': // press e to exit
      tidy_and_exit();
      break;

    default:
      printf("\nPress escape or 'e' 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);

  int size, rank;

  MPI_Init(NULL, NULL);
  MPI_Comm_size(MPI_COMM_WORLD, &size);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if(size != 5) {
    if(rank == 0) {
      printf("This program needs 5 processes\n");
    }
  } else {
    if(rank ==0){
            struct timespec start, finish;
       long long int time_elapsed;
       clock_gettime(CLOCK_MONOTONIC, &start);
            MPI_Send(&results[0], 1800, MPI_UNSIGNED_CHAR, 1, 0,
MPI_COMM_WORLD);
            MPI_Send(&results[1800], 1800, MPI_UNSIGNED_CHAR, 2, 0,
MPI_COMM_WORLD);
            MPI_Send(&results[3600], 1800, MPI_UNSIGNED_CHAR, 3, 0,
MPI_COMM_WORLD);
            MPI_Send(&results[5400], 1800, MPI_UNSIGNED_CHAR, 4, 0,
MPI_COMM_WORLD);

            MPI_Recv(&results[0], 1800, MPI_UNSIGNED_CHAR, 1, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            MPI_Recv(&results[1800], 1800,MPI_UNSIGNED_CHAR, 2, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            MPI_Recv(&results[3600], 1800,MPI_UNSIGNED_CHAR, 3, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            MPI_Recv(&results[5400], 1800,MPI_UNSIGNED_CHAR, 4, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);

            clock_gettime(CLOCK_MONOTONIC, &finish);
             time_difference(&start, &finish, &time_elapsed);
            printf("Time elapsed was %lldns or %0.9lfs\n",
time_elapsed,(time_elapsed/1.0e9));

             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();


    } else {
      if(rank == 1){

              first = 0;
             last = 1799;

             MPI_Recv(&results[0], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
             detect_edges(image, results);
             MPI_Send(&results[0], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD);
      }
       else if(rank == 2){
             first = 1800;
             last = 3599;

             MPI_Recv(&results[1800], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
             detect_edges(image, results);
             MPI_Send(&results[1800], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD);
       }
       else if(rank == 3){
              first = 3600;
              last = 5399;

              MPI_Recv(&results[3600], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
              detect_edges(image, results);
               MPI_Send(&results[3600], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD);

       }
       else if(rank == 4){
              first = 5400;
              last = 7199;

              MPI_Recv(&results[5400], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD, MPI_STATUS_IGNORE);
              detect_edges(image, results);
              MPI_Send(&results[5400], 1800, MPI_UNSIGNED_CHAR, 0, 0,
MPI_COMM_WORLD);
```

```
        }
      }
    }
  MPI_Finalize();
  return 0;
}

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

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  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,
  255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,
  0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,255,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,0,255,255,0,0,255,255,255,
  255,255,255,0,255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,
  255,255,255,255,0,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  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,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  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,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
  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,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
  255,255,255,0,255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,255,
  0,0,255,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,0,
  0,0,0,255,255,0,0,0,255,0,0,0,0,255,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  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,
  0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
  0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,255,255,
  255,255,255,0,0,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,255,255,0,0,0,0,0,255,255,0,0,0,0,
  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,

0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  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,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
  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,
  0,255,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,
  0,0,0,255,0,0,0,255,0,0,0,0,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,255,255,255,0,0,255,255,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,255,0,255,255,255,255,255,255,
  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,
  255,255,255,255,255,0,0,0,0,0,0,255,255,255,255,0,255,255,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,255,

255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,25
5,
  255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,255,255,255,255,255,0,0,0,255,255,255,0,255,255,0,
  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,
  0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,0,0,0,255,255,
  255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
  0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
  255,0,0,255,255,255,255,255,255,255,255,255,0,255,0,0,0,0,0
};
```
Illustration 40: image processing with mpi

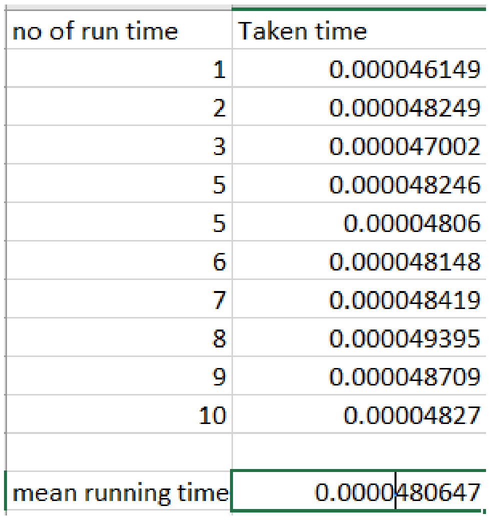
b) Compare the mean running time of the MPI version with the original, multithread and CUDA versions.

Answer:



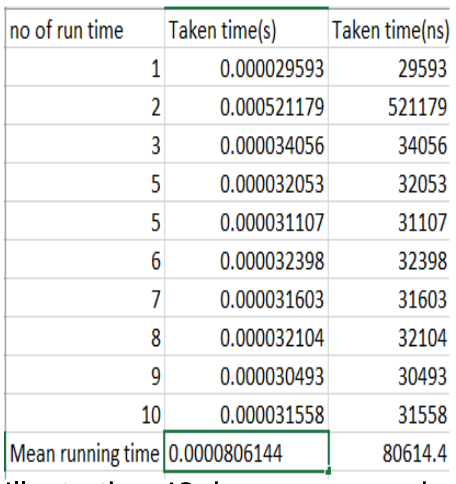
no of run time	Taken time
1	0.00016907
2	0.000248894
3	0.000218004
5	0.000143842
5	0.000337346
6	0.000152765
7	0.000171662
8	0.000182074
9	0.000131864
10	0.000156094
Mean running time	0.0001911615

Illustration 41: image processing (original)

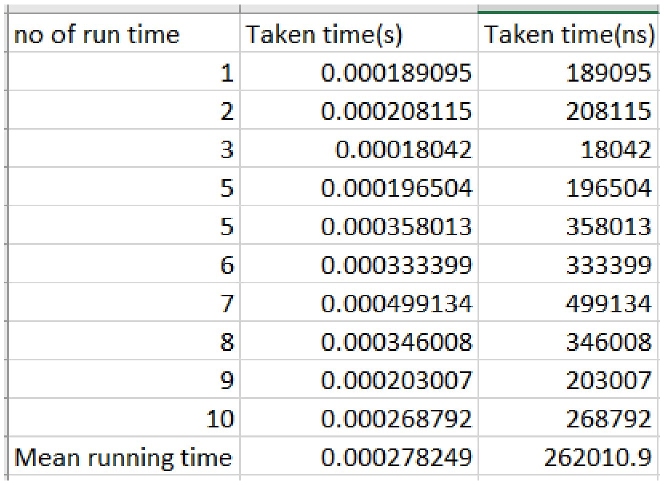


no of run time	Taken time
1	0.000046149
2	0.000048249
3	0.000047002
5	0.000048246
5	0.00004806
6	0.000048148
7	0.000048419
8	0.000049395
9	0.000048709
10	0.00004827
mean running time	0.0000480647

Illustration 42: image processing with posix thread



no of run time	Taken time(s)	Taken time(ns)
1	0.000029593	29593
2	0.000521179	521179
3	0.000034056	34056
5	0.000032053	32053
5	0.000031107	31107
6	0.000032398	32398
7	0.000031603	31603
8	0.000032104	32104
9	0.000030493	30493
10	0.000031558	31558
Mean running time	0.0000806144	80614.4



no of run time	Taken time(s)	Taken time(ns)
1	0.000189095	189095
2	0.000208115	208115
3	0.00018042	18042
5	0.000196504	196504
5	0.000358013	358013
6	0.000333399	333399
7	0.000499134	499134
8	0.000346008	346008
9	0.000203007	203007
10	0.000268792	268792
Mean running time	0.000278249	262010.9

Illustration 43: image processing with Cuda

Illustration 43:image processing with mpi

4.2.1 MPI Image Processing Code Analysis:

The main aim of this code is to know how to computer stores and manipulates two-dimensional array of cells called image. Picture made up of pixels which made up of horizontal and vertical coordinated x and y. Two time_spec start and finish along with the time elapsed is declared in declaration phase to access the start time, finish time and the time difference respectively. Clock_get time function is used to get the time from the system which takes two parameters CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time. MPI library is initialized by MPI_Init datatype. MPI_Comm_Size() function is used to identify size of communicator.MPI_Comm_rank() function is used to identify the rank of the calling process.

4.3 Linear regression:

a) Create another version of the linear regression program that uses 8 MPI instances, to each compute the error associated with a specific regression line. The master instance should run the main algorithm and use a different instance to compute the error associate with each m, c being explored.

Answer:

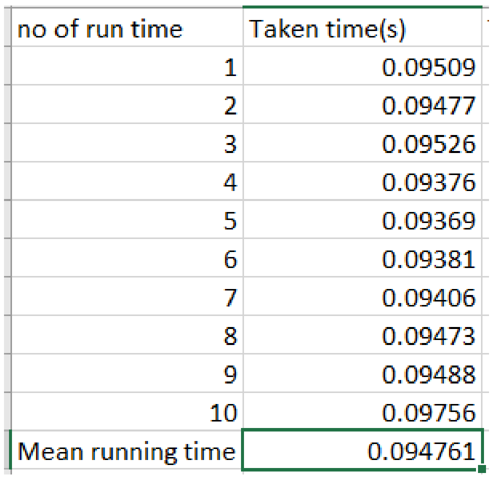
Source code of linear regression with mpi:

```
#include <stdio.h>
#include <math.h>
#include <mpi.h>
#include <time.h>

/*********************************************************************
*
* To compile:
*   mpicc -o linearRegressionwith_mpi linearRegressionwith_mpi.c -lm
*
* To run:
*   mpirun -n 9 ./linearRegressionwith_mpi
**********************************************************************
***/
```

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

int n_data = 1000;
point_t data[];

double 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);
}
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);

   int rank, size;
   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 min_found = 0;
   double error_p = 0;
   //double base_mc[2];

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


   MPI_Init (NULL, NULL);
   MPI_Comm_size (MPI_COMM_WORLD, &size);
   MPI_Comm_rank (MPI_COMM_WORLD, &rank);

   be = rms_error (bm, bc);

   if (size != 9)
   {
      if (rank == 0)
      {
         printf
            ("This program is designed to run with exactly 9
processes.\n");
         return 0;
      }
   }

   while (!min_found)
   {

      if (rank != 0)
      {
         i = rank - 1;
         dm[i] = bm + (om[i] * step);
         dc[i] = bc + (oc[i] * step);
         error_p = rms_error (dm[i], dc[i]);

         MPI_Send (&error_p, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
         MPI_Send (&dm[i], 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);
         MPI_Send (&dc[i], 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD);


         MPI_Recv (&bm, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
         MPI_Recv (&bc, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
         MPI_Recv (&min_found, 1, MPI_INT, 0, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
      }
      else
      {
         for (i = 1; i < size; i++)
         {
            MPI_Recv (&error_p, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            MPI_Recv (&dm[i-1], 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            MPI_Recv (&dc[i-1], 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            if (error_p < best_error)
            {
               best_error = error_p;
               best_error_i = i - 1;

            }
         }
         // printf ("best m,c is %lf,%lf with error %lf in direction
%d\n",
         // dm[best_error_i], dc[best_error_i], best_error, best_error_i);
         if (best_error < be)
         {
            be = best_error;
            bm = dm[best_error_i];
            bc = dc[best_error_i];
         }
         else
         {
            min_found = 1;
         }

         for (i = 1; i < size; i++)
         {
            MPI_Send (&bm, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
            MPI_Send (&bc, 1, MPI_DOUBLE, i, 0, MPI_COMM_WORLD);
            MPI_Send (&min_found, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
         }
      }
   }

   if(rank==0) {
      printf ("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);
      clock_gettime(CLOCK_MONOTONIC, &finish);
      time_difference(&start, &finish, &time_elapsed);
      printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
         (time_elapsed/1.0e9));
   }

   MPI_Finalize();
   return 0;
}

point_t data[] = {
  {65.11,126.40},{76.79,149.00},{76.93,162.00},{65.24,113.46},
  {78.54,145.93},{84.60,161.77},{85.60,162.58},{82.32,152.21},
  {78.17,144.80},{69.47,142.78},{82.72,156.05},{11.56,52.20},
  {66.15,122.01},{75.13,145.75},{ 8.11,36.01},{71.58,150.44},
  {23.30,70.06},{42.59,86.42},{39.11,76.86},{ 8.77,36.29},
  {83.41,152.32},{ 3.44,36.86},{72.15,126.11},{66.29,129.15},
  {28.93,92.24},{91.62,172.01},{ 0.39,40.02},{55.24,104.88},
  {44.96,90.98},{89.66,170.29},{29.39,86.66},{56.19,109.96},
  {79.43,153.15},{54.27,110.73},{ 9.28,54.91},{31.16,76.74},
  {20.00,49.29},{67.25,122.68},{95.64,182.45},{66.03,128.30},
  {36.60,94.60},{83.93,120.16},{22.67,76.02},{81.17,164.59},
  {84.70,147.15},{34.58,87.81},{ 0.26,39.25},{82.07,149.43},
  { 2.63,41.39},{ 1.74, 7.95},{70.98,133.50},{16.65,49.48},
  {27.85,61.85},{55.84,105.63},{81.77,153.60},{19.81,61.26},
  {28.19,97.35},{ 2.62,32.52},{60.42,123.43},{53.67,118.83},
  {92.67,163.43},{ 4.09,30.06},{31.35,78.55},{54.79,103.97},
  {89.15,163.38},{20.35,66.02},{28.55,88.62},{11.66,58.29},
  {89.90,154.40},{ 0.14,51.92},{ 4.75,37.69},{53.83,108.99},
  {62.17,127.68},{79.10,133.64},{24.19,68.78},{51.41,100.86},
  {44.52,92.93},{23.02,66.51},{98.60,181.12},{ 6.05,48.82},
  {62.79,147.70},{ 5.06,50.58},{85.40,155.28},{12.33,60.17},
  {49.62,118.33},{ 9.03,48.29},{45.21,88.73},{28.22,55.37},
  {91.32,165.67},{ 6.74,44.19},{46.03,93.83},{69.75,139.69},
  { 2.15,40.31},{95.82,160.20},{ 6.64,54.91},{75.25,148.74},
  {39.64,68.97},{ 5.55,66.26},{90.53,155.37},{39.95,91.42},
  {68.89,132.98},{33.52,78.37},{15.84,38.51},{72.73,139.50},
  {21.54,73.78},{ 4.64,47.34},{66.57,132.87},{27.38,71.86},
  {93.83,181.33},{75.83,161.75},{26.47,56.70},{84.23,151.43},
  { 0.43,43.29},{88.50,160.27},{66.15,129.59},{78.31,141.68},
  {36.90,101.61},{71.78,139.52},{90.37,173.79},{ 0.58,45.98},
  {67.63,131.85},{57.43,100.37},{88.43,161.15},{74.83,132.98},
  {29.31,54.66},{79.06,146.78},{54.41,120.80},{51.76,108.96},
  {11.80,65.51},{38.19,90.48},{18.40,71.77},{76.29,148.07},
  {75.30,135.15},{59.56,126.34},{32.71,86.25},{42.35,116.15},
  { 4.85,38.50},{ 3.14,50.60},{48.27,90.59},{34.96,88.02},
  {10.03,50.01},{ 5.51,40.83},{68.32,136.16},{74.87,134.02},
  { 1.56,47.50},{19.52,72.68},{ 9.10,52.12},{50.79,102.10},
  {53.11,105.38},{94.93,174.68},{16.03,44.26},{13.26,49.58},
  { 3.24,46.86},{77.38,158.65},{21.57,62.81},{41.63,89.86},
  {13.55,59.72},{25.43,71.35},{86.73,166.79},{77.15,149.52},
  {26.47,64.94},{48.65,92.62},{33.66,75.10},{25.20,63.45},
  {25.46,86.18},{70.52,147.39},{98.32,175.47},{23.09,62.68},
  {48.90,118.74},{69.07,141.45},{50.54,132.25},{55.80,119.88},
  {25.65,92.01},{54.39,112.81},{79.86,165.28},{95.98,154.86},
  {48.14,108.06},{36.33,88.43},{ 6.34,35.96},{86.04,151.77},
  {57.03,116.32},{97.95,180.12},{29.66,73.83},{12.52,35.04},
  {43.93,83.56},{33.63,78.69},{64.00,128.13},{14.49,50.14},
  {49.66,112.89},{82.54,162.20},{81.92,143.19},{28.07,78.90},
  {14.26,47.92},{23.97,63.31},{27.69,73.01},{78.13,119.54},
  {34.43,82.48},{66.13,123.89},{61.84,135.81},{17.03,57.30},
  { 5.61,52.51},{34.44,88.49},{17.81,81.52},{34.26,79.71},
  {93.17,161.29},{ 8.10,39.44},{93.51,158.23},{61.48,133.51},
  {27.22,71.93},{17.11,50.22},{27.73,81.68},{16.07,61.27},
  {63.81,122.63},{ 0.27,36.83},{62.21,120.74},{42.36,85.01},
  {60.61,143.23},{68.59,121.10},{28.48,68.27},{23.39,71.50},
  {93.40,162.60},{50.72,114.87},{24.53,80.83},{92.00,160.38},
  {79.29,175.12},{28.84,78.42},{13.79,44.14},{23.18,62.24},
  {69.07,122.23},{41.93,120.63},{62.32,125.07},{72.39,136.08},
  {41.86,92.41},{ 2.35,21.43},{56.14,133.02},{33.91,90.07},
  {13.68,76.01},{14.55,71.51},{73.79,152.07},{33.47,97.28},
  {31.12,65.68},{ 4.33,41.19},{22.94,58.10},{85.12,160.37},
  {80.26,154.39},{37.01,78.54},{ 6.94,38.10},{ 7.83,60.51},
  {42.44,90.12},{26.69,91.63},{99.36,184.47},{ 9.33,55.05},
  {16.87,63.97},{32.41,80.49},{36.34,77.15},{59.91,122.62},
  { 7.35,34.97},{99.21,183.69},{34.07,97.06},{43.85,102.96},
  {55.20,111.99},{ 3.95,41.47},{26.71,82.57},{16.69,64.61},
  {38.32,96.85},{76.67,136.21},{86.22,175.61},{35.18,71.39},
  {57.39,117.85},{72.12,139.30},{90.19,173.32},{97.26,163.25},
  {82.08,135.04},{40.69,96.78},{25.49,75.76},{83.38,149.39},
  {63.64,135.54},{90.52,166.25},{79.96,154.68},{45.70,107.42},
  {15.54,61.07},{97.10,170.63},{41.10,87.33},{35.86,74.41},
  {57.22,120.65},{16.28,64.80},{46.33,97.53},{31.84,83.82},
  {90.15,177.85},{13.39,77.75},{ 8.25,26.83},{91.74,155.44},
  {11.65,61.09},{26.30,82.75},{61.72,128.34},{76.94,152.59},
  {26.70,81.25},{ 6.11,39.57},{97.46,172.22},{13.50,43.37},
  {16.46,54.81},{44.36,84.30},{45.83,105.17},{41.47,105.60},
  {31.72,82.66},{58.79,113.04},{95.35,168.80},{27.91,73.77},
  {61.28,126.49},{ 1.18,32.24},{ 4.17,41.67},{79.08,142.98},
  { 4.80,37.58},{94.98,160.79},{37.47,80.15},{ 6.82,53.58},
  {57.54,118.51},{73.31,139.05},{91.40,166.67},{98.07,162.54},
  { 3.87,41.53},{63.71,130.41},{75.78,137.34},{56.32,122.22},
  { 8.03,63.52},{22.60,60.09},{94.56,158.12},{16.10,72.60},
  {82.22,155.28},{57.63,114.94},{55.26,118.38},{73.52,126.54},
  {59.13,123.52},{81.11,167.56},{68.73,123.16},{43.78,122.00},
  {27.48,70.82},{43.92,110.09},{ 7.04,44.70},{91.03,147.13},
  {44.55,114.71},{68.40,122.77},{ 6.31,55.15},{12.03,51.80},
  {26.62,77.09},{12.90,60.85},{41.47,115.00},{75.98,156.05},
  {62.84,118.68},{ 3.19,54.74},{74.93,132.72},{89.37,170.57},
  {57.12,114.25},{63.07,104.90},{60.20,129.41},{36.04,82.56},
  {66.43,133.16},{ 7.01,45.98},{87.68,162.25},{36.24,86.35},
  {60.38,140.39},{ 3.56,41.80},{65.74,138.56},{16.34,61.32},
  {34.00,105.97},{77.42,132.46},{61.79,135.32},{61.13,116.51},
  {90.22,159.21},{68.78,137.53},{48.35,110.41},{58.48,110.06},
  { 0.04,45.54},{87.14,152.92},{73.42,154.90},{48.73,105.35},
  {36.26,97.55},{50.34,107.46},{95.65,162.87},{11.76,50.19},
  {12.83,58.44},{ 7.59,65.35},{51.44,109.31},{15.39,65.30},
  {83.69,147.00},{75.86,139.91},{25.87,87.89},{ 0.79,40.68},
  { 4.87,62.14},{64.71,126.73},{60.94,111.46},{82.18,142.90},
  {21.67,55.08},{33.20,93.76},{11.93,64.07},{10.59,39.64},
  {20.90,63.85},{21.47,81.20},{15.02,80.95},{67.04,112.88},
  { 0.78,34.00},{24.49,77.44},{ 0.49,33.31},{70.88,138.01},
  {33.07,81.36},{74.11,139.95},{ 1.26,40.24},{26.68,81.03},
  {81.37,155.39},{89.28,163.35},{82.92,150.44},{88.94,166.64},
  {14.35,63.72},{58.92,119.69},{47.88,100.06},{73.51,145.81},
  {21.11,79.13},{33.98,87.52},{87.88,158.32},{24.47,79.36},
  {34.00,92.04},{42.25,105.99},{20.75,64.71},{65.45,131.84},
  {10.51,38.23},{36.26,87.53},{70.72,129.61},{62.87,129.51},
  {88.93,170.73},{50.72,124.30},{52.28,104.85},{ 1.38,36.72},
  {93.31,171.99},{56.94,135.14},{99.80,174.15},{ 2.74,46.10},
  {42.39,81.82},{42.07,102.65},{ 3.06,34.94},{69.46,129.37},
  {43.33,104.41},{69.47,129.06},{68.70,134.00},{95.28,179.59},
  {66.66,132.99},{14.40,70.02},{39.22,77.48},{ 5.87,32.85},
  {71.70,144.19},{69.40,140.87},{51.33,113.42},{38.58,97.94},
  {70.18,129.30},{28.53,76.70},{11.77,52.53},{16.26,69.96},
  {86.39,161.92},{14.35,60.74},{17.32,53.69},{60.10,117.59},
  {55.86,117.63},{ 9.26,58.48},{48.28,100.74},{79.81,149.90},
  {59.38,105.17},{72.31,119.55},{41.23,91.88},{70.20,136.75},
  {82.58,158.00},{72.29,130.76},{10.80,60.44},{81.60,167.47},
  {57.21,120.91},{83.97,150.98},{78.97,152.79},{78.71,146.61},
  {98.28,172.82},{39.89,87.34},{92.46,169.18},{29.94,69.32},
  {64.78,127.52},{52.32,113.01},{29.54,78.81},{46.04,97.77},
  {97.71,155.81},{90.08,172.72},{59.58,117.24},{74.61,149.80},
  {64.20,116.45},{42.14,102.85},{44.27,107.95},{97.48,174.75},
  {84.52,164.71},{19.46,51.01},{87.05,166.73},{28.47,61.63},
  { 0.59,40.82},{49.83,100.11},{25.05,62.68},{20.32,62.69},
  {56.64,126.50},{15.41,59.79},{98.36,173.55},{57.73,97.55},
  {29.88,86.10},{26.44,65.10},{92.31,174.76},{10.74,49.39},
  {88.32,163.82},{71.67,156.47},{94.40,181.01},{16.04,55.25},
  {76.78,141.43},{81.76,146.31},{81.41,144.40},{ 7.59,65.42},
  {29.83,93.19},{71.85,128.96},{17.45,45.91},{66.78,148.29},
  {87.12,149.00},{55.37,115.72},{ 8.34,69.56},{10.30,58.08},
  {26.49,79.04},{33.65,83.88},{82.84,146.92},{74.19,132.28},
  {47.38,105.49},{95.30,172.52},{ 8.60,54.66},{38.14,69.96},
  {14.06,65.00},{27.26,74.14},{31.28,82.60},{24.83,97.76},
  {53.38,101.53},{26.88,87.40},{77.79,153.42},{ 4.60,53.07},
  {36.70,76.53},{33.96,93.76},{64.04,116.47},{73.85,156.81},
  {31.88,85.18},{10.44,62.68},{41.55,99.48},{31.20,94.01},
  {69.63,137.46},{30.90,92.46},{54.24,103.71},{82.12,149.86},
  {57.43,119.86},{16.83,61.60},{38.45,74.99},{ 9.38,60.12},
  {18.91,53.31},{65.28,115.98},{52.45,119.66},{ 4.88,52.30},
  {49.92,95.81},{60.44,126.35},{25.07,83.25},{58.21,108.57},
  {28.81,94.01},{44.94,99.55},{35.75,79.36},{95.26,164.51},
  {35.14,113.38},{ 6.95,51.77},{37.56,81.49},{27.79,72.55},
  {68.04,129.74},{19.46,82.90},{13.49,40.02},{40.81,92.01},
  {44.13,86.46},{90.16,178.62},{82.34,153.55},{92.32,165.64},
  {78.20,166.78},{24.76,66.00},{91.00,175.56},{85.94,172.81},
  {98.74,178.91},{ 7.47,34.02},{37.28,85.95},{ 8.94,54.67},
  {31.78,85.85},{31.71,82.87},{44.29,96.83},{21.85,68.96},
  {15.20,48.69},{ 9.51,58.55},{14.53,53.46},{87.25,166.09},
  {35.25,79.77},{45.43,106.79},{16.24,69.34},{61.36,142.83},
  {99.33,176.18},{ 2.66,42.92},{42.69,105.85},{69.04,124.32},
  {62.77,125.38},{87.76,149.94},{68.38,124.70},{44.95,109.62},
  { 8.36,63.51},{29.47,75.02},{42.49,87.92},{29.05,95.14},
  { 1.36,43.70},{60.36,102.16},{23.57,54.88},{30.84,80.74},
  {10.19,42.03},{97.59,177.44},{36.08,89.31},{21.74,45.86},
  {58.56,113.61},{34.10,92.54},{87.76,174.79},{43.42,107.45},
  {55.01,110.06},{45.87,119.35},{21.24,61.64},{ 0.63,23.13},
  {44.94,99.54},{ 5.22,47.01},{ 1.71,42.19},{92.32,159.09},
  {28.15,76.89},{77.98,128.92},{40.11,84.47},{80.44,144.10},
  {21.62,80.78},{27.18,70.12},{80.83,148.92},{65.54,132.52},
  {69.13,124.43},{26.54,59.95},{ 0.13,36.97},{24.07,70.64},
  {27.58,70.42},{45.07,121.14},{11.82,46.41},{81.39,156.60},
  {49.46,95.96},{56.25,93.87},{92.93,167.21},{85.35,169.34},
  {32.46,93.55},{37.88,93.61},{66.98,144.61},{67.21,133.07},
  {37.90,81.47},{68.35,136.90},{69.28,140.78},{78.26,143.36},
  {28.73,69.02},{48.85,91.09},{ 6.11,61.37},{69.24,156.76},
  {58.32,123.43},{13.23,59.97},{32.85,74.58},{48.15,120.84},
  {74.60,145.21},{46.64,104.61},{63.37,120.76},{13.36,59.46},
  {69.89,142.17},{67.89,136.10},{49.22,99.19},{73.16,143.29},
  {47.79,130.64},{41.71,94.63},{93.46,171.77},{99.74,185.80},
  {58.15,112.90},{24.90,82.06},{17.53,58.51},{34.06,80.58},
  {51.11,115.72},{19.12,64.68},{29.05,80.50},{30.71,91.87},
  {20.00,77.43},{38.82,97.86},{25.56,71.28},{24.69,51.78},
  {15.15,52.31},{89.92,178.26},{97.21,171.26},{54.16,134.74},
  {84.67,149.81},{74.93,123.09},{ 5.26,24.90},{99.04,183.04},
  {89.69,180.72},{ 9.57,59.78},{27.52,86.77},{ 7.79,63.47},
  {86.70,159.99},{12.54,49.44},{65.80,139.16},{60.68,99.45},
  {37.01,99.10},{65.32,128.72},{79.27,139.94},{13.48,59.51},
  {16.15,65.81},{ 5.50,56.27},{21.44,61.06},{17.95,80.39},
  {22.99,69.66},{78.04,139.81},{ 8.19,45.53},{53.04,114.50},
  {22.03,55.53},{71.11,134.99},{12.41,60.57},{47.53,107.37},
  { 0.20,27.63},{ 3.31,26.26},{59.81,132.51},{50.17,104.10},
  {84.25,141.14},{91.89,171.66},{14.95,62.25},{ 9.00,61.06},
  {29.68,75.94},{98.55,169.15},{63.95,127.72},{41.36,82.03},
  {92.20,168.84},{71.55,142.74},{89.17,168.56},{36.19,84.82},
  { 5.83,58.93},{32.71,82.95},{13.63,72.12},{20.78,69.59},
  {96.66,156.89},{40.74,93.92},{12.50,64.55},{91.70,165.65},
  {45.68,89.74},{10.70,52.42},{80.60,159.09},{46.91,99.34},
  {42.30,97.16},{34.03,85.62},{68.84,132.20},{94.47,166.73},
  { 6.57,23.33},{88.09,172.72},{10.29,44.01},{16.28,64.39},
  {40.21,82.53},{42.50,101.48},{85.18,145.73},{88.49,176.79},
  {23.93,69.17},{21.58,71.42},{43.56,101.34},{18.85,72.03},
  { 4.01,20.86},{58.74,130.89},{ 0.55,42.23},{64.01,138.48},
  {86.32,164.34},{ 4.01,62.96},{71.65,145.59},{59.98,128.80},
  {47.29,107.25},{52.80,112.62},{73.48,143.42},{60.71,105.76},
  {14.39,46.36},{91.65,166.65},{68.70,134.37},{17.20,63.05},
  {49.86,111.33},{15.66,66.77},{13.85,55.13},{11.74,62.94},
  {46.11,92.86},{90.43,144.43},{12.80,46.53},{ 8.49,48.78},
  {92.34,176.52},{77.18,145.53},{18.95,72.13},{25.16,77.45},
  {79.17,156.72},{94.54,168.51},{12.56,52.73},{31.32,80.71},
  {83.67,145.98},{69.02,141.65},{16.67,51.28},{43.22,108.69},
  { 2.77,44.28},{28.19,92.70},{85.57,161.86},{16.23,62.41},
  { 7.59,70.56},{36.61,85.26},{31.17,83.60},{77.49,151.10},
  {12.82,38.79},{30.11,81.59},{50.07,122.10},{74.50,144.63},
  {94.48,175.21},{82.49,146.39},{47.18,90.69},{19.81,68.22},
  {67.87,135.07},{86.53,158.63},{ 4.02,67.70},{79.22,163.68},
  {18.63,65.68},{93.39,170.96},{95.97,163.34},{75.47,121.35},
  { 0.78,37.11},{ 9.53,50.40},{39.13,110.03},{95.69,168.67},
  {27.61,84.96},{47.10,120.52},{96.66,178.29},{88.15,179.79},
  {54.08,127.28},{98.67,173.36},{28.33,79.71},{ 3.98,31.32},
  {98.84,179.12},{22.71,70.55},{ 2.25,35.21},{32.51,72.10},
  {61.33,121.66},{70.04,137.97},{47.57,129.83},{15.27,63.70},
  {67.47,148.68},{90.29,162.66},{ 5.58,56.85},{26.24,75.05},
  {97.20,190.42},{97.93,174.98},{72.40,139.24},{36.57,100.59},
  { 9.55,69.48},{28.55,80.48},{23.97,69.20},{40.40,94.65},
  {93.43,169.59},{56.99,101.50},{29.82,78.34},{63.85,105.18},
  {36.57,93.67},{29.99,100.46},{48.09,99.60},{17.13,85.66},
  {42.67,102.26},{26.34,76.52},{ 9.81,48.84},{35.70,76.14},
  {89.40,153.75},{97.80,177.01},{27.89,69.25},{46.43,113.97},
  {21.64,62.84},{72.79,131.04},{86.23,150.89},{57.53,122.30},
  {36.87,91.32},{13.15,50.63},{81.13,165.31},{29.36,108.50},
  {25.65,81.54},{21.91,58.02},{60.06,128.34},{53.90,116.97},
  {37.20,91.22},{ 2.75,54.02},{78.84,143.43},{78.42,129.08},
  {30.30,91.45},{ 6.19,51.64},{15.94,75.49},{49.50,107.77},
  {48.58,103.97},{42.05,114.46},{98.55,169.49},{23.59,77.96},
  { 4.95,31.04},{51.61,122.22},{89.57,166.43},{97.29,183.00},
  {67.36,143.50},{70.70,143.79},{ 7.09,61.57},{ 4.55,35.73},
  { 3.12,26.08},{27.61,71.71},{17.87,65.37},{73.82,148.35},
  {71.86,152.17},{39.75,97.64},{11.52,51.38},{84.82,150.22},
  {33.13,77.12},{34.83,83.95},{53.84,105.93},{85.86,161.20},
  {80.36,135.81},{29.48,66.91},{33.44,84.75},{27.94,89.60},
  {61.89,130.52},{15.65,50.50},{66.84,126.11},{61.89,124.02},
  {30.64,82.56},{63.67,113.67},{93.79,175.50},{89.78,180.21},
  {49.60,106.06},{78.60,152.09},{88.82,171.67},{ 4.49,41.76},
  {12.41,62.47},{57.54,122.63},{42.00,96.54},{15.89,62.16},
  {18.09,43.62},{98.19,177.35},{49.84,105.13},{59.38,128.63},
  {55.34,118.20},{60.21,125.47},{31.34,69.51},{79.20,139.77},
  {26.37,81.64},{45.32,72.27},{91.13,173.22},{91.36,169.43},
  {65.10,128.76},{24.33,59.90},{39.37,93.39},{88.88,156.65},
  {66.86,146.50},{73.40,126.02},{14.09,64.93},{87.34,173.83},
  {18.26,68.89},{92.26,160.92},{77.91,157.56},{52.89,98.98},
  {38.14,109.31},{41.50,96.53},{26.81,89.59},{47.42,103.41},
  {68.58,132.42},{60.29,126.09},{64.99,125.45},{76.35,144.33},
  {11.69,57.61},{28.16,72.44},{23.94,72.18},{95.67,182.61},
  {59.17,118.32},{35.19,83.30},{19.53,74.53},{45.16,96.72},
  {66.63,128.79},{96.13,182.09},{65.31,126.98},{33.27,102.77},
  { 3.65,39.52},{19.26,74.36},{32.61,70.26},{37.77,82.99},
  { 1.77,32.37},{87.50,167.27},{90.60,158.93},{86.81,154.12},
  {23.83,77.55},{97.47,166.55},{83.99,167.80},{44.51,104.49},
  {86.46,168.85},{75.17,142.50},{83.71,173.31},{92.83,162.93}
};
```
Illustration 44: Linear regression with mpi program

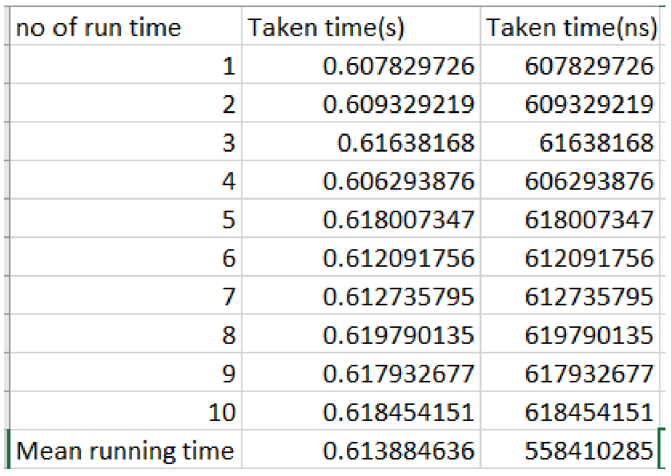
b) Compare the mean running time of the MPI version with the original, multithread and CUDA versions.

Ans:



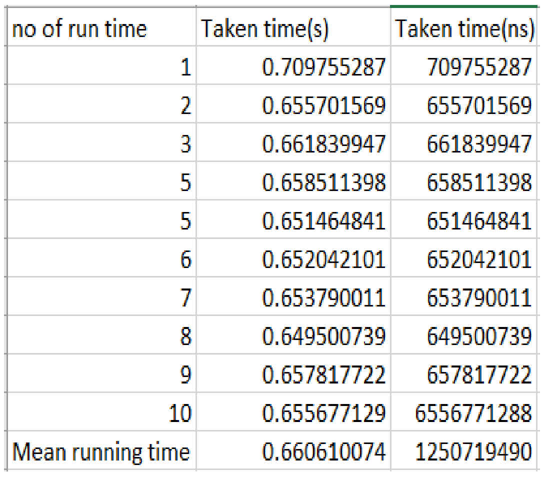
no of run time	Taken time(s)
1	0.09509
2	0.09477
3	0.09526
4	0.09376
5	0.09369
6	0.09381
7	0.09406
8	0.09473
9	0.09488
10	0.09756
Mean running time	0.094761

Illustration 47:Linear regression Original program



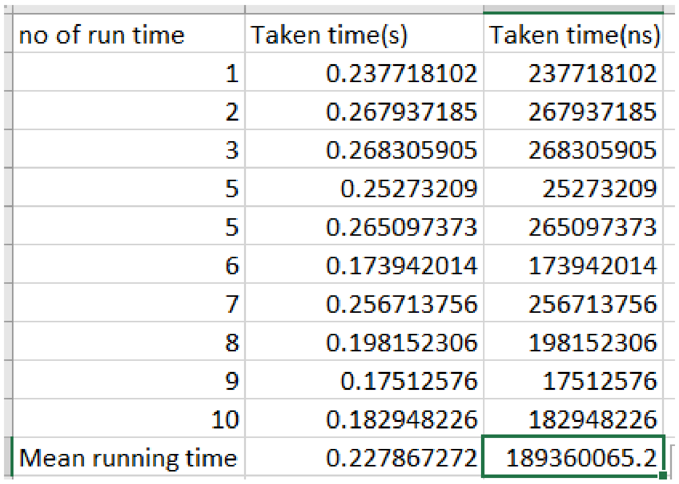
no of run time	Taken time(s)	Taken time(ns)
1	0.607829726	607829726
2	0.609329219	609329219
3	0.61638168	61638168
4	0.606293876	606293876
5	0.618007347	618007347
6	0.612091756	612091756
7	0.612735795	612735795
8	0.619790135	619790135
9	0.617932677	617932677
10	0.618454151	618454151
Mean running time	0.613884636	558410285

Illustration 47:Linear regression with posix thread



no of run time	Taken time(s)	Taken time(ns)
1	0.709755287	709755287
2	0.655701569	655701569
3	0.661839947	661839947
5	0.658511398	658511398
5	0.651464841	651464841
6	0.652042101	652042101
7	0.653790011	653790011
8	0.649500739	649500739
9	0.657817722	657817722
10	0.655677129	6556771288
Mean running time	0.660610074	1250719490

Illustration 47:Linear regression with cuda



no of run time	Taken time(s)	Taken time(ns)
1	0.237718102	237718102
2	0.267937185	267937185
3	0.268305905	268305905
5	0.25273209	25273209
5	0.265097373	265097373
6	0.173942014	173942014
7	0.256713756	256713756
8	0.198152306	198152306
9	0.17512576	17512576
10	0.182948226	182948226
Mean running time	0.227867272	189360065.2

Illustration 48: Linear regression with mpi

4.3.1 MPI Linear Regression Code Analysis:

The main object of this code to find out the minimum value m,c and error. To calculate time two time-spec start and finish and finish along with elapsed is declared in declaration phase to access the start time , finish time and the time difference. Clock_get time function is used to get the time from the system which takes two parameters CLOCK_MONOTONIC and &start or CLOCK_MONOTONIC and & finish to get the start declare to variable size and time. MPI library is initialized by MPI_Init datatype. MPI_Comm_Size() function is used to identify size of communicator.MPI_Comm_rank() function is used to identify the rank of the calling process.