6CS005 Learning Journal - Semester 1 2019/20

Put your name and student number here

Table of Contents

Table of Contents 1

1 POSIX Threads 2

1.1 Password Cracking 2

1.2 Image Processing 12

1.3 Linear Regression 27

2 CUDA 43

2.1 Password Cracking 43

2.2 Image Processing 43

2.3 Linear Regression 43

3 MPI 44

3.1 Password Cracking 44

3.2 Image Processing 44

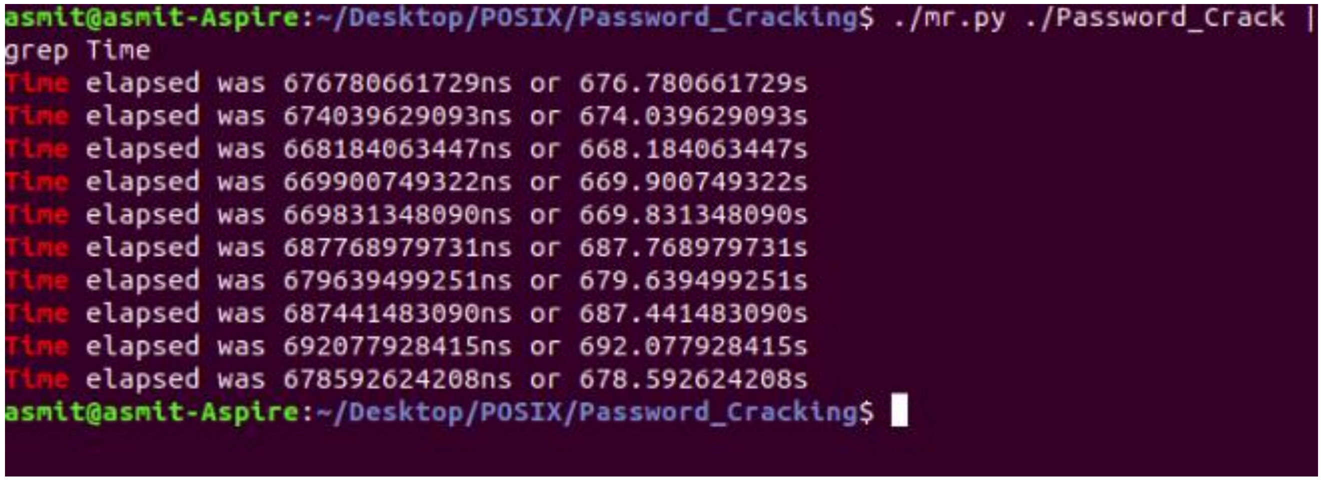
3.3 Linear Regression 44

4 Verbose Repository Log 45

1 POSIX Threads

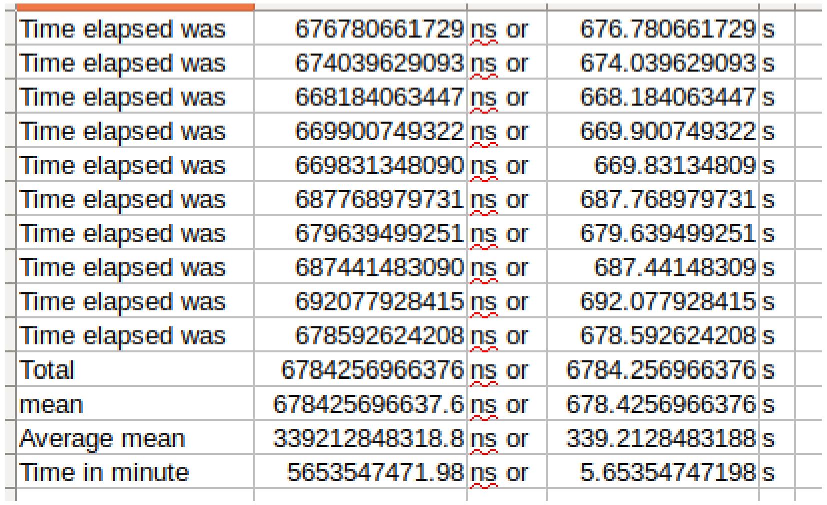
1.1 Password Cracking

Insert a table of 10 running times and the mean running time.



```
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking$ ./mr.py ./Password_Crack |
grep Time
Time elapsed was 676780661729ns or 676.780661729s
Time elapsed was 674039629093ns or 674.039629093s
Time elapsed was 668184063447ns or 668.184063447s
Time elapsed was 669900749322ns or 669.900749322s
Time elapsed was 669831348090ns or 669.831348090s
Time elapsed was 687768979731ns or 687.768979731s
Time elapsed was 679639499251ns or 679.639499251s
Time elapsed was 687441483090ns or 687.441483090s
Time elapsed was 692077928415ns or 692.077928415s
Time elapsed was 678592624208ns or 678.592624208s
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking$
```

Figure 1 Running password cracking program for 10 time



Time elapsed was	676780661729	ns or	676.780661729	s
Time elapsed was	674039629093	ns or	674.039629093	s
Time elapsed was	668184063447	ns or	668.184063447	s
Time elapsed was	669900749322	ns or	669.900749322	s
Time elapsed was	669831348090	ns or	669.83134809	s
Time elapsed was	687768979731	ns or	687.768979731	s
Time elapsed was	679639499251	ns or	679.639499251	s
Time elapsed was	687441483090	ns or	687.44148309	s
Time elapsed was	692077928415	ns or	692.077928415	s
Time elapsed was	678592624208	ns or	678.592624208	s
Total	6784256966376	ns or	6784.256966376	s
mean	678425696637.6	ns or	678.4256966376	s
Average mean	339212848318.8	ns or	339.2128483188	s
Time in minute	5653547471.98	ns or	5.65354747198	s

Figure 2 calculating total, mean and average mean time in csv file

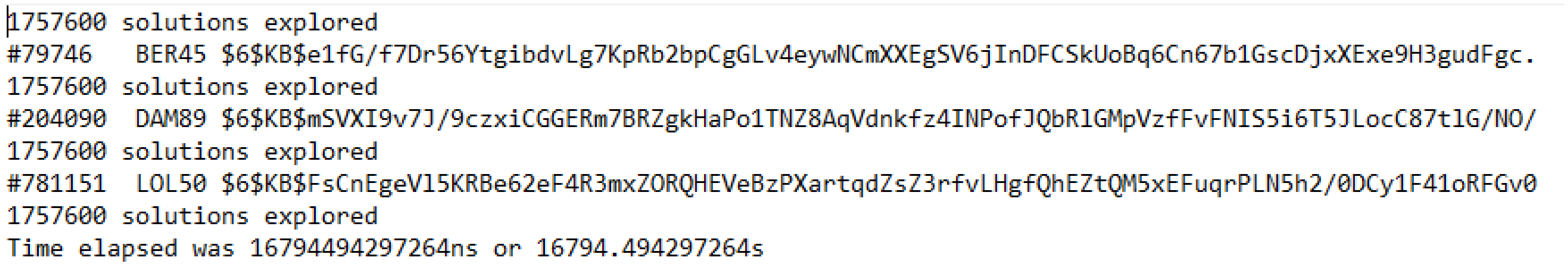
Insert a paragraph that hypothesises how long it would take to run if the number of initials were to be increased to 3. Include your calculations.

- It would take more time than two initial if the number of initials were to be increased to 3. From figure:2 it is clear that it takes about 6 minute to run two initial program but if the initial were increased it is obvious to increase in time as the program will go through each and every for loops. It would take 26 time more if the initial were to be increased to 3 that means it will take about 6*26 =(156 min) time for completing 3 initial program.

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



int n_passwords = 4;

 char *encrypted_passwords[] = {
   "$6$KB$6S9dCIrK1jRJu0AoISAn1nWkUEnmDAGm/sKs1Vb1GJXhSrRJv0pe9jyRvM9ohg96sSQt9IPNPC6obgGTl6P2A1",
   "$6$KB$e1fG/f7Dr56YtgibdvLg7KpRb2bpCgGLv4eywNCmXXEgSV6jInDFCSkUoBq6Cn67b1GscDjxXExe9H3gudFgc.",
   "$6$KB$mSVXI9v7J/9czxiCGGERm7BRZgkHaPo1TNZ8AqVdnkfz4INPofJQbRlGMpVzfFvFNIS5i6T5JLocC87tlG/NO/",
   "$6$KB$FsCnEgeVl5KRBe62eF4R3mxZORQHEVeBzPXartqdZsZ3rfvLHgfQhEZtQM5xEFuqrPLN5h2/0DCy1F41oRFGv0"
 };

 /**
  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';
 }

 /**
  This function can crack the kind of password explained above. All combinations
  that are tried are displayed and when the password is found, #, is put at the
  start of the line. Note that one of the most time consuming operations that
  it performs is the output of intermediate results, so performance experiments
  for this kind of program should not include this. i.e. comment out the printfs.
 */

 void Passwordcrack(char *salt_and_encrypted){
   int x, y, z, a;      // 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='A'; z<='Z'; z++){
         for(a=0; a<=99; a++){
                sprintf(plain, "%c%c%c%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);
 }

 //Calculating time

 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++)
        {
                Passwordcrack(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;
```

Explain your results of running your 3 initial password cracker with relation to your earlier hypothesis.



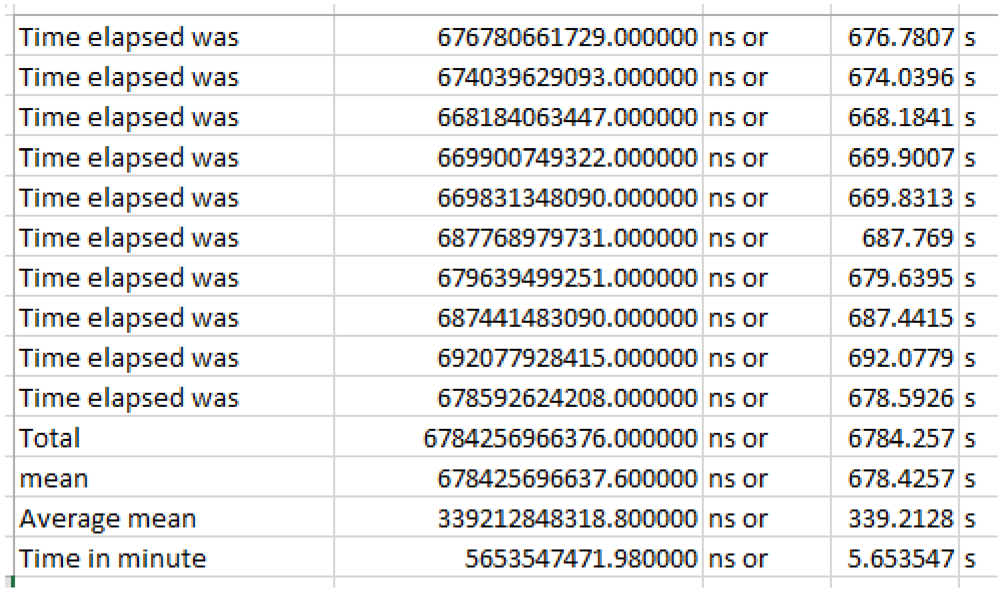
```
1757600 solutions explored
#79746   BER45 $6$KB$e1fG/f7Dr56YtgibdvLg7KpRb2bpCgGLv4eywNCmXXEgSV6jInDFCSkUoBq6Cn67b1GscDjxXExe9H3gudFgc.
1757600 solutions explored
#204090  DAM89 $6$KB$mSVXI9v7J/9czxiCGGERm7BRZgkHaPo1TNZ8AqVdnkfz4INPofJQbRlGMpVzfFvFNIS5i6T5JLocC87tlG/NO/
1757600 solutions explored
#781151  LOL50 $6$KB$FsCnEgeVl5KRBe62eF4R3mxZORQHEVeBzPXartqdZsZ3rfvLHgfQhEZtQM5xEFuqrPLN5h2/0DCy1F41oRFGv0
1757600 solutions explored
Time elapsed was 16794494297264ns or 16794.494297264s
```

-

Figure 3 3 initial password cracking

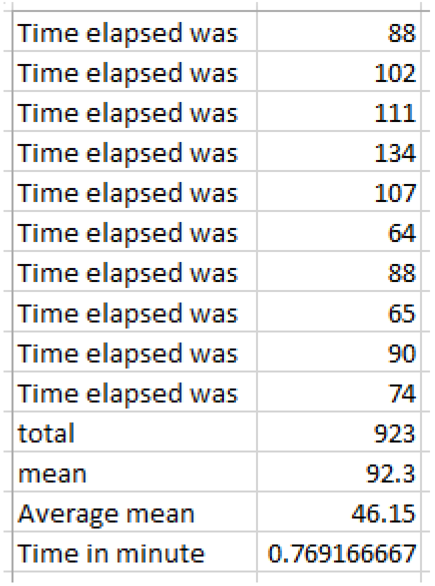
In my hypothesis I said that it will take about 156 min to complete 3 initial program but it took about 279 minute (16794.50 sec).

Write a paragraph that compares the original results with those of your multithread password cracker.



Time elapsed was	676780661729.000000	ns or	676.7807	s
Time elapsed was	674039629093.000000	ns or	674.0396	s
Time elapsed was	668184063447.000000	ns or	668.1841	s
Time elapsed was	669900749322.000000	ns or	669.9007	s
Time elapsed was	669831348090.000000	ns or	669.8313	s
Time elapsed was	687768979731.000000	ns or	687.769	s
Time elapsed was	679639499251.000000	ns or	679.6395	s
Time elapsed was	687441483090.000000	ns or	687.4415	s
Time elapsed was	692077928415.000000	ns or	692.0779	s
Time elapsed was	678592624208.000000	ns or	678.5926	s
Total	6784256966376.000000	ns or	6784.257	s
mean	678425696637.600000	ns or	678.4257	s
Average mean	339212848318.800000	ns or	339.2128	s
Time in minute	5653547471.980000	ns or	5.653547	s

Figure 4 Calculating mean time of simple program



Time elapsed was	88
Time elapsed was	102
Time elapsed was	111
Time elapsed was	134
Time elapsed was	107
Time elapsed was	64
Time elapsed was	88
Time elapsed was	65
Time elapsed was	90
Time elapsed was	74
total	923
mean	92.3
Average mean	46.15
Time in minute	0.769166667

Figure 5 Calculating mean time of multithread program

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

int n_passwords = 4;

 char *encrypted_passwords[] = {
 "$6$KB$0G24VuNaA9ApVG4z8LkI/OOr9a54nBfzgQjbebhqBZxMHNg0HiYYf1Lx/HcGg6q1nnOSArPtZYbGy7yc5V.wP/",
 "$6$KB$VDUCASt5S88l82JzexhKDQLeUJ5zfxr16VhlVwNOs0YLiLYDciLDmN3QYAE80UIzfryYmpR.NFmbZvAGNoaHW.",
 "$6$KB$0n1YjoLnJBuAdeBsYFW3fpZzMPP8xycQbEj35GvoerMnEkWIAKnbUBAb70awv5tfHylWkVzcwzHUNy/7l7I1c/",
 "$6$KB$HKffNNiGzngqYueF89z3gwWZMg.xUBIz/00QSCbgwKtRHmwUbZX6jTH4VUAg3L3skaO8qtNf5LE7WP39jQ7ZJ0"
 };


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

```

/**
 This function can crack the kind of password explained above. All
combinations
 that are tried are displayed and when the password is found, #, is put
at the
 start of the line. Note that one of the most time consuming operations
that
 it performs is the output of intermediate results, so performance
experiments
 for this kind of program should not include this. i.e. comment out the
printfs.
*/

void posix()
{
  int i;
pthread_t thread1, thread2;

    void *kernel_function_1();
    void *kernel_function_2();
for(i=0;i<n_passwords;i<i++) {


    pthread_create(&thread1, NULL,kernel_function_1, encrypted_passwords[i]);
    pthread_create(&thread2, NULL,kernel_function_2, encrypted_passwords[i]);

    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);

 }
}

void *kernel_function_1(char *salt_and_encrypted){
  int x, y, z;      // Loop counters
  char salt[7];
  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++){
        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);
        }
      }
    }
  }
  printf("%d solutions explored\n", count);
}

void *kernel_function_2(char *salt_and_encrypted){
  int i, j, k;      // Loop counters
  char salt[7];     // String used in hahttps://www.youtube.com/watch?v=L8yJjIGleMwshing 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(i='N'; i<='Z'; i++){
    for(j='A'; j<='Z'; j++){
      for(k=0; k<=99; k++){
        sprintf(plain, "%c%c%02d", i,j,k);
        enc = (char *) crypt(plain, salt);
        count++;
        if(strcmp(salt_and_encrypted, enc) == 0){
          printf("#%-8d%s %s\n", count, plain, enc);
        }
      }
    }
  }
  printf("%d solutions explored\n", count);
}

//Calculating time

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;
        posix();
        clock_gettime(CLOCK_MONOTONIC, &start);
        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;
}
```

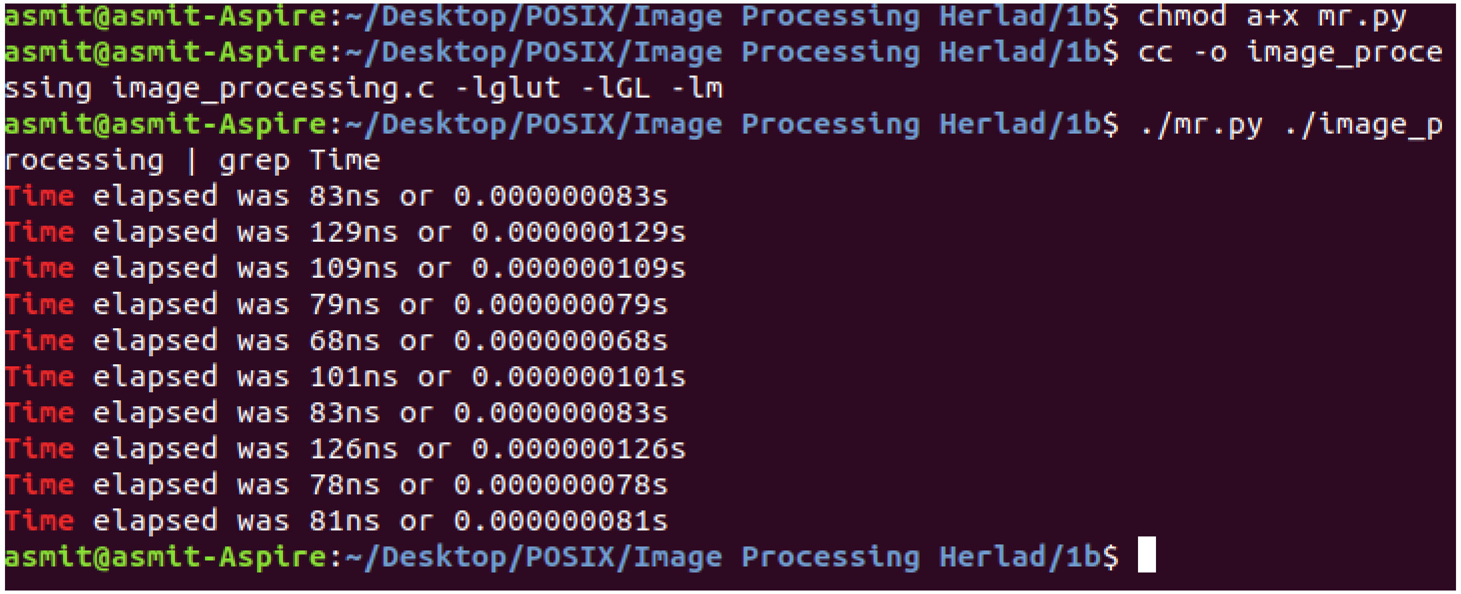
1.2 Image Processing

Insert the image displayed by your program

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

 #define width 100
 #define height 72
 typedef struct arg_t{
 int start;
 int stride;
 }arg_t;

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

 void edges(unsigned char *in, unsigned char *out,arg_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;
```

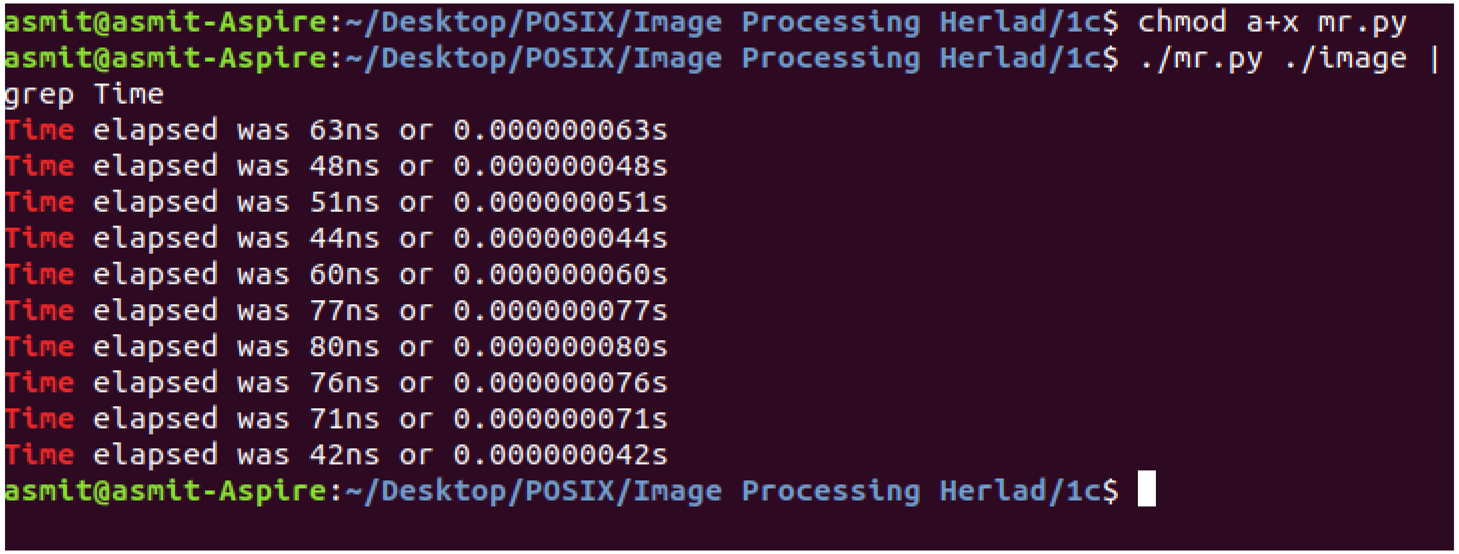
```

        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 *Detection(void *args)
{
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 show() {
  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);
   glutInit(&argc, argv);
  struct timespec start, finish;
    long long int time_elapsed;

    clock_gettime(CLOCK_MONOTONIC, &start);



    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));

    printf("image dimensions %dx%d\n", width, height);
   pthread_t t1, t2, t3, t4;

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

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

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

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

  void *Detection();

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

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



  glutInitWindowSize(width * 2,height);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_LUMINANCE);

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

  glutMainLoop();

  tidy_and_exit();
 pthread_exit(&t1);
  pthread_exit(&t2);
  pthread_exit(&t3);
  pthread_exit(&t4);

  return 0;
}

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

Insert a table that has columns containing running times for the original program and your multithread version. Mean running times should be included at the bottom of the columns.



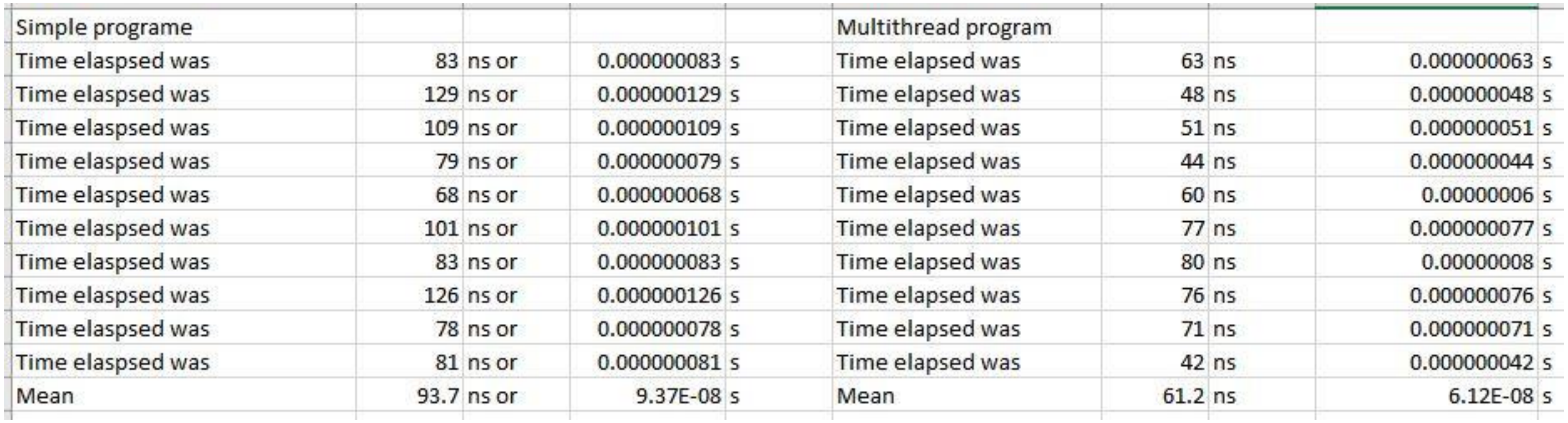
```
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1b$ chmod a+x mr.py
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1b$ cc -o image_proce
ssing image_processing.c -lglut -lGL -lm
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1b$ ./mr.py ./image_p
rocessing | grep Time
Time elapsed was 83ns or 0.000000083s
Time elapsed was 129ns or 0.000000129s
Time elapsed was 109ns or 0.000000109s
Time elapsed was 79ns or 0.000000079s
Time elapsed was 68ns or 0.000000068s
Time elapsed was 101ns or 0.000000101s
Time elapsed was 83ns or 0.000000083s
Time elapsed was 126ns or 0.000000126s
Time elapsed was 78ns or 0.000000078s
Time elapsed was 81ns or 0.000000081s
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1b$ 
```

Figure 6 Running simple program of image processing for 10 time



```
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1c$ chmod a+x mr.py
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1c$ ./mr.py ./image |
grep Time
Time elapsed was 63ns or 0.000000063s
Time elapsed was 48ns or 0.000000048s
Time elapsed was 51ns or 0.000000051s
Time elapsed was 44ns or 0.000000044s
Time elapsed was 60ns or 0.000000060s
Time elapsed was 77ns or 0.000000077s
Time elapsed was 80ns or 0.000000080s
Time elapsed was 76ns or 0.000000076s
Time elapsed was 71ns or 0.000000071s
Time elapsed was 42ns or 0.000000042s
asmit@asmit-Aspire:~/Desktop/POSIX/Image Processing Herlad/1c$
```

Figure 7 Running multithread program of image processing for 10 time



Simple programe				Multithread program				
Time elaspsed was	83	ns or	0.000000083	s	Time elapsed was	63	ns	0.000000063	s
Time elaspsed was	129	ns or	0.000000129	s	Time elapsed was	48	ns	0.000000048	s
Time elaspsed was	109	ns or	0.000000109	s	Time elapsed was	51	ns	0.000000051	s
Time elaspsed was	79	ns or	0.000000079	s	Time elapsed was	44	ns	0.000000044	s
Time elaspsed was	68	ns or	0.000000068	s	Time elapsed was	60	ns	0.00000006	s
Time elaspsed was	101	ns or	0.000000101	s	Time elapsed was	77	ns	0.000000077	s
Time elaspsed was	83	ns or	0.000000083	s	Time elapsed was	80	ns	0.00000008	s
Time elaspsed was	126	ns or	0.000000126	s	Time elapsed was	76	ns	0.000000076	s
Time elaspsed was	78	ns or	0.000000078	s	Time elapsed was	71	ns	0.000000071	s
Time elaspsed was	81	ns or	0.000000081	s	Time elapsed was	42	ns	0.000000042	s
Mean	93.7	ns or	9.37E-08	s	Mean	61.2	ns	6.12E-08	s

Figure 8 mean time of simple program vs multithread program

Insert an explanation of the results presented in the above table.

1.3 Linear Regression

Insert a scatter plot of your data.

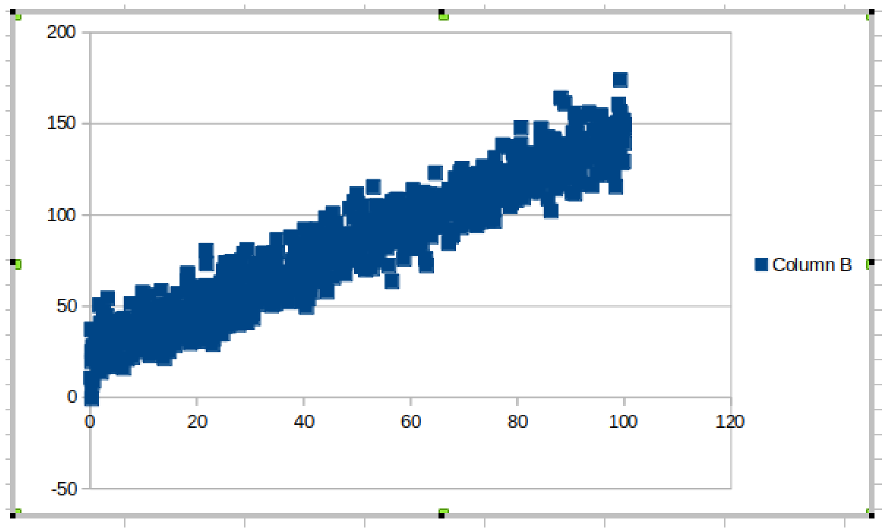
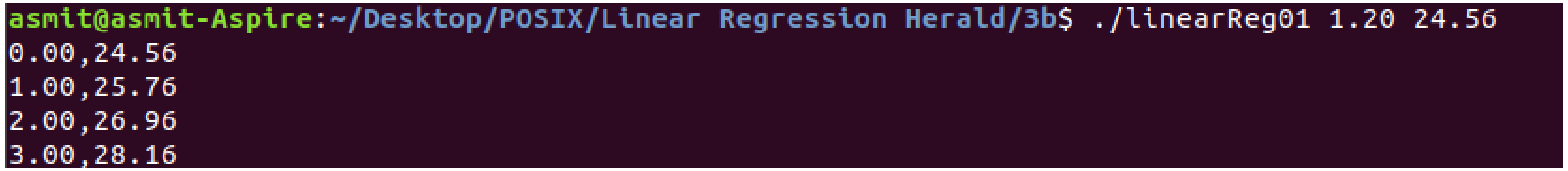


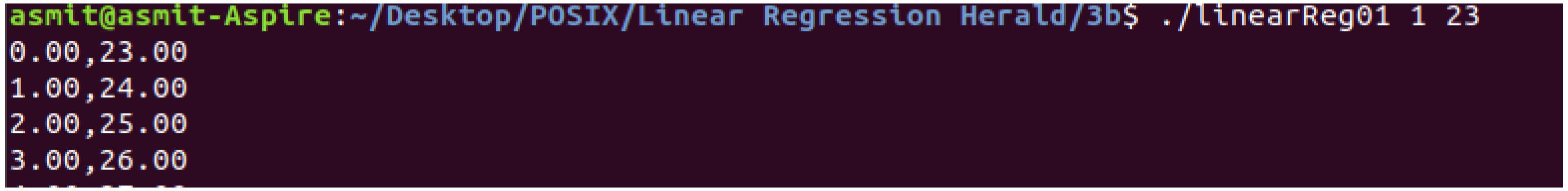
Figure 9 Scatter plot

Have 3 guesses at the optimum values for m and c and present them in a graph that overlays your data.



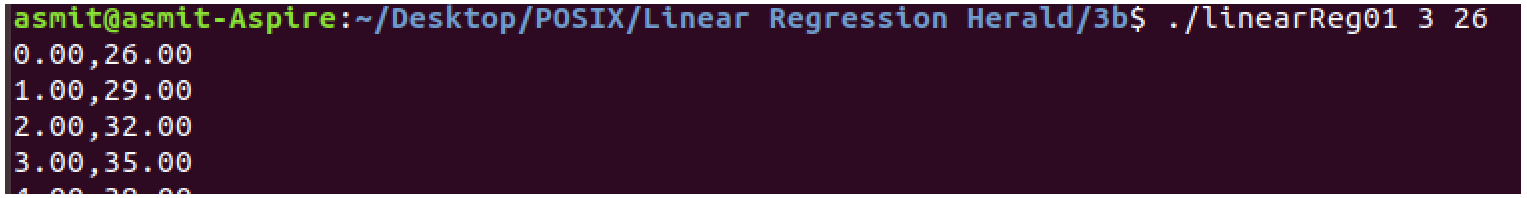
```
asmit@asmit-Aspire:~/Desktop/POSIX/Linear Regression Herald/3b$ ./linearReg01 1.20 24.56
0.00,24.56
1.00,25.76
2.00,26.96
3.00,28.16
```

Figure 10 m=1.20 and c=24.56



```
asmit@asmit-Aspire:~/Desktop/POSIX/Linear Regression Herald/3b$ ./linearReg01 1 23
0.00,23.00
1.00,24.00
2.00,25.00
3.00,26.00
```

Figure 11 m=1 and c=23



```
asmit@asmit-Aspire:~/Desktop/POSIX/Linear Regression Herald/3b$ ./linearReg01 3 26
0.00,26.00
1.00,29.00
2.00,32.00
3.00,35.00
```

Figure 12 m=3 and c=26

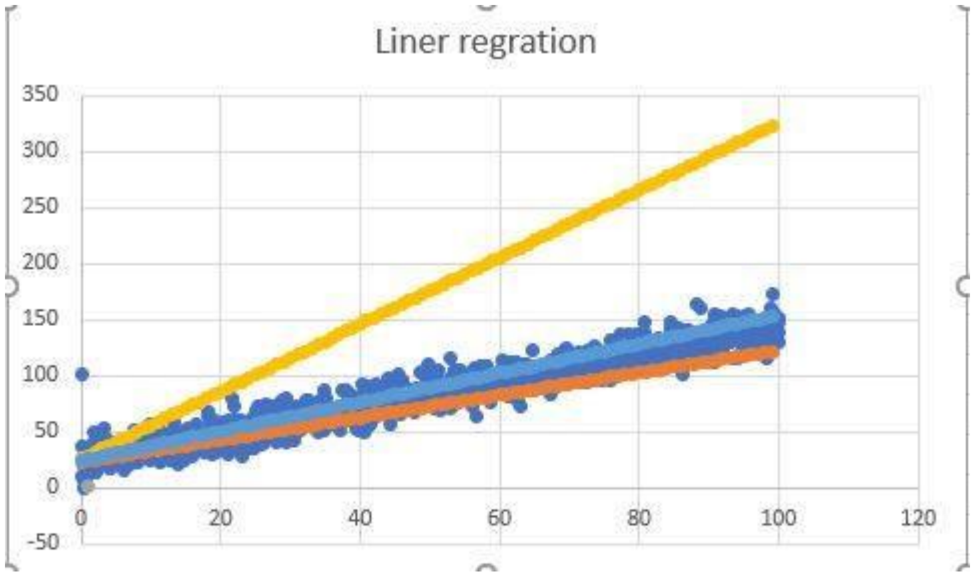
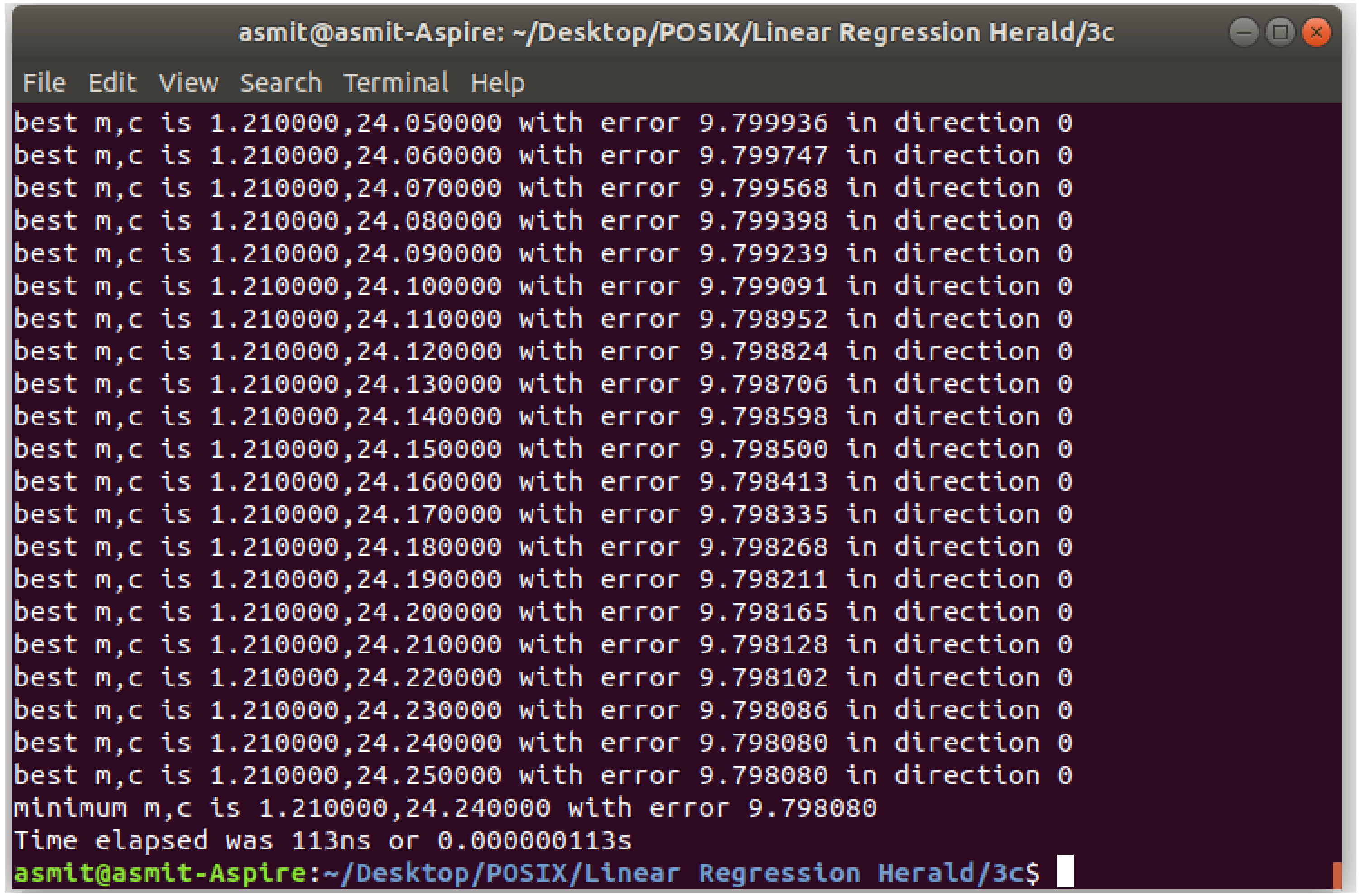


Figure 13 overlaying 3 guesses of m and c over scatter data

Insert a graph that presents your data with the solution overlaid.


asmit@asmit-Aspire: ~/Desktop/POSIX/Linear Regression Herald/3c
File Edit View Search Terminal Help

```
best m,c is 1.210000,24.050000 with error 9.799936 in direction 0
best m,c is 1.210000,24.060000 with error 9.799747 in direction 0
best m,c is 1.210000,24.070000 with error 9.799568 in direction 0
best m,c is 1.210000,24.080000 with error 9.799398 in direction 0
best m,c is 1.210000,24.090000 with error 9.799239 in direction 0
best m,c is 1.210000,24.100000 with error 9.799091 in direction 0
best m,c is 1.210000,24.110000 with error 9.798952 in direction 0
best m,c is 1.210000,24.120000 with error 9.798824 in direction 0
best m,c is 1.210000,24.130000 with error 9.798706 in direction 0
best m,c is 1.210000,24.140000 with error 9.798598 in direction 0
best m,c is 1.210000,24.150000 with error 9.798500 in direction 0
best m,c is 1.210000,24.160000 with error 9.798413 in direction 0
best m,c is 1.210000,24.170000 with error 9.798335 in direction 0
best m,c is 1.210000,24.180000 with error 9.798268 in direction 0
best m,c is 1.210000,24.190000 with error 9.798211 in direction 0
best m,c is 1.210000,24.200000 with error 9.798165 in direction 0
best m,c is 1.210000,24.210000 with error 9.798128 in direction 0
best m,c is 1.210000,24.220000 with error 9.798102 in direction 0
best m,c is 1.210000,24.230000 with error 9.798086 in direction 0
best m,c is 1.210000,24.240000 with error 9.798080 in direction 0
best m,c is 1.210000,24.250000 with error 9.798080 in direction 0
minimum m,c is 1.210000,24.240000 with error 9.798080
Time elapsed was 113ns or 0.000000113s
asmit@asmit-Aspire:~/Desktop/POSIX/Linear Regression Herald/3c$
```

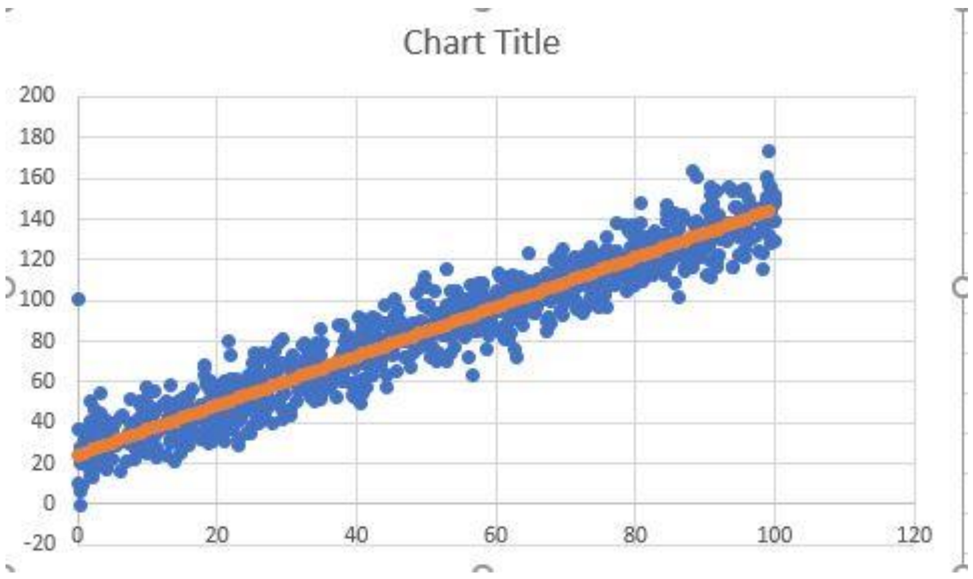
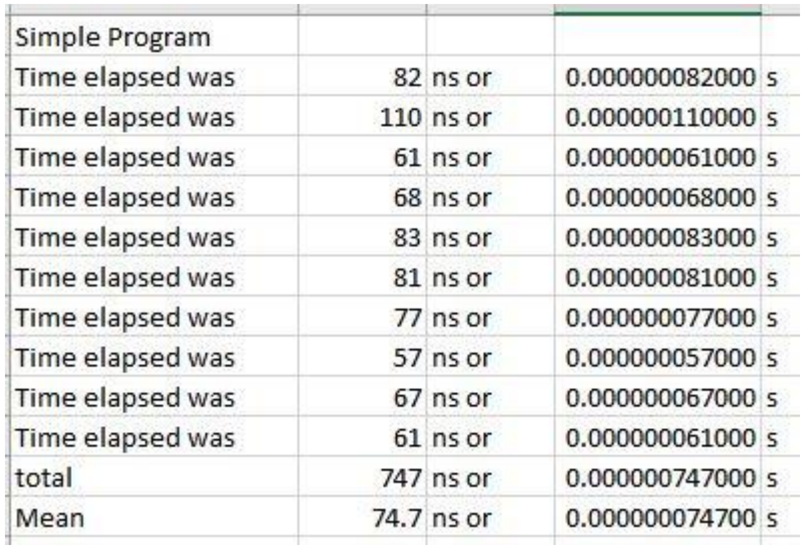


Figure 14 solution of data give where m=1.21 and c=24.24

Insert a comment that compares your guesses with the solution found.

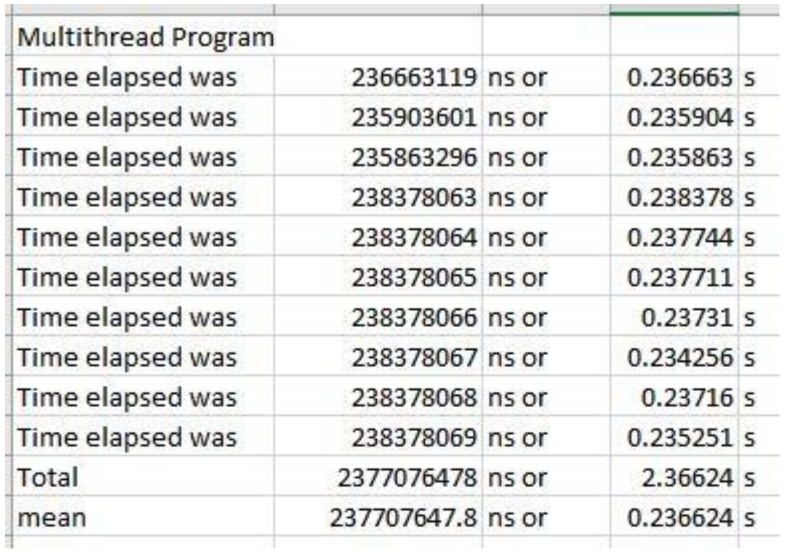
```c
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <pthread.h>
#include <string.h>
#include <stdlib.h>


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};

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(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,linear_regression,&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[] = {
  {72.12,100.78},{65.40,107.86},{82.27,131.60},{82.31,122.34},
  {89.41,121.50},{71.37,113.51},{82.62,112.38},{69.57,102.96},
  {65.38,99.27},{84.50,138.85},{87.18,114.17},{73.03,109.21},
  {67.26,102.06},{72.25,113.23},{61.28,101.59},{41.60,84.24},
  {40.14,57.03},{15.24,45.58},{61.88,89.90},{34.89,72.77},
  { 8.91,36.34},{30.45,46.18},{67.93,89.35},{68.82,112.80},
  {63.96,99.32},{32.36,56.12},{42.20,63.66},{24.47,60.75},
  { 1.96,28.62},{41.42,68.41},{34.49,73.14},{ 8.03,22.13},
  {80.55,117.79},{85.54,130.80},{68.99,103.13},{99.32,144.79},
{91.71,153.61},{71.17,108.40},{85.28,120.11},{99.52,128.68},
{13.24,31.67},{ 5.19,40.15},{ 9.84,57.36},{29.42,54.01},
{89.68,126.25},{29.45,41.30},{79.63,132.59},{71.88,107.31},
{20.05,48.38},{40.98,54.11},{56.55,63.61},{77.22,114.17},
{63.86,88.10},{92.93,134.84},{56.84,101.20},{34.31,71.18},
{93.89,116.43},{38.02,63.78},{61.25,94.71},{71.02,103.42},
{95.05,142.82},{96.24,133.50},{19.50,50.92},{41.14,70.59},
{91.49,134.05},{54.05,98.31},{36.59,68.48},{91.14,130.45},
{44.76,88.98},{77.28,138.16},{64.80,96.33},{43.25,70.08},
{55.55,95.70},{ 3.77,39.03},{ 3.23,44.69},{86.72,127.42},
{84.62,131.54},{26.13,71.24},{61.22,98.22},{53.90,96.07},
{64.81,109.35},{91.66,116.79},{53.65,104.81},{38.42,66.16},
{62.33,112.41},{ 7.41,29.86},{41.59,57.59},{56.49,91.60},
{15.94,42.82},{97.46,140.29},{57.17,85.11},{26.94,45.86},
{73.14,96.37},{18.61,60.58},{15.69,44.16},{20.79,33.86},
{65.02,106.03},{38.09,72.71},{87.15,116.68},{77.45,123.08},
{90.47,126.33},{26.80,44.96},{75.94,119.76},{33.83,69.11},
{63.59,103.98},{38.05,72.36},{68.28,110.76},{ 3.34,54.22},
{45.40,92.84},{78.37,113.49},{27.11,46.46},{32.32,68.44},
{20.97,30.90},{37.92,75.11},{96.85,130.96},{69.40,95.17},
{ 3.29,30.06},{64.41,103.44},{15.80,52.64},{61.76,97.79},
{ 1.62,33.98},{29.03,58.02},{18.74,34.93},{25.41,73.73},
{28.78,65.94},{14.64,50.31},{82.85,133.70},{41.62,90.32},
{99.28,144.95},{90.16,133.18},{40.45,77.72},{ 1.79,50.44},
{31.80,62.71},{26.30,40.89},{47.57,83.15},{17.78,44.90},
{69.48,93.13},{87.98,126.95},{69.84,106.00},{37.06,61.61},
{90.65,133.97},{10.73,46.60},{38.84,79.90},{ 4.75,33.89},
{48.99,89.31},{ 2.51,47.09},{34.99,86.40},{29.79,54.52},
{91.30,133.72},{74.12,122.86},{90.93,141.88},{51.14,89.93},
{84.53,142.49},{26.84,58.79},{ 6.95,20.98},{49.80,85.14},
{22.82,57.02},{44.08,89.32},{22.28,48.72},{21.12,50.68},
{65.69,93.93},{27.84,39.97},{ 1.92,40.39},{ 9.36,33.54},
{88.10,123.02},{18.15,63.84},{21.80,39.76},{64.42,101.03},
{ 2.23,22.52},{55.68,99.56},{37.55,87.77},{74.23,104.87},
{11.96,37.30},{23.60,45.84},{11.13,34.32},{ 9.05,48.79},
{56.11,100.21},{19.31,54.44},{ 6.27,16.17},{64.65,101.39},
{50.25,77.59},{69.33,95.12},{47.52,87.79},{28.97,65.98},
{71.56,95.30},{19.71,41.47},{57.66,96.65},{41.07,74.10},
{35.08,79.46},{40.80,87.01},{ 0.31,19.82},{90.78,111.55},
{34.39,72.03},{99.97,139.40},{30.86,73.03},{14.37,50.15},
{ 6.11,42.76},{21.75,80.30},{89.94,127.56},{10.86,42.40},
{13.07,42.98},{84.47,147.14},{83.44,132.18},{32.24,63.57},
{66.93,102.41},{34.48,68.96},{ 3.46,22.82},{94.84,130.83},
{49.41,107.26},{71.64,99.82},{47.28,80.62},{39.17,68.77},
{58.05,108.35},{69.27,109.81},{47.64,73.34},{34.64,73.15},
{22.86,46.34},{37.76,66.19},{ 3.12,39.11},{60.59,111.05},
{91.99,122.76},{96.60,138.86},{ 3.58,23.35},{22.81,60.18},
{13.93,21.32},{69.51,106.41},{19.57,43.39},{79.11,115.68},
{80.89,124.36},{44.42,57.78},{33.28,73.04},{21.45,49.88},
{70.57,113.77},{45.63,65.60},{55.99,72.21},{21.62,41.47},
{61.74,98.99},{ 9.30,29.77},{75.32,106.74},{27.97,73.44},
{74.77,115.98},{42.93,82.67},{92.32,138.05},{25.55,64.34},
{ 0.48,23.51},{79.52,111.52},{52.83,70.58},{51.45,87.28},
{62.72,90.41},{ 4.16,40.60},{70.13,115.25},{55.96,97.34},
{93.88,154.09},{46.21,90.04},{34.75,51.46},{54.45,89.56},
{80.69,129.36},{45.14,73.00},{47.34,85.69},{70.16,118.02},
{ 4.26,17.14},{61.56,98.04},{15.95,28.56},{74.06,118.48},
{65.29,99.71},{19.08,55.64},{37.82,72.36},{58.22,103.93},
{50.52,82.15},{26.25,60.91},{97.77,123.91},{39.13,68.03},
{15.09,41.88},{32.61,61.64},{11.23,22.85},{61.92,98.02},
{73.63,126.32},{35.12,54.74},{12.98,42.69},{83.87,128.60},
{45.65,78.81},{42.85,90.57},{76.74,117.53},{19.05,49.60},
{69.03,104.16},{23.66,54.97},{52.85,85.94},{82.07,128.27},
{74.77,111.22},{95.04,136.69},{40.49,49.53},{ 4.16,28.40},
{ 7.69,51.29},{29.37,80.82},{86.06,122.19},{ 3.92,23.24},
{62.76,108.89},{27.12,54.24},{10.24,33.84},{79.86,107.97},
{57.09,85.27},{10.29,54.38},{53.50,82.98},{12.83,50.29},
{ 2.09,13.69},{88.73,135.16},{42.72,87.10},{40.20,91.88},
{40.10,76.49},{80.22,133.65},{57.55,93.99},{29.34,69.08},
{ 2.90,41.26},{44.60,82.03},{47.93,89.05},{98.17,123.11},
{17.21,45.91},{42.37,79.83},{90.89,119.42},{ 7.81,36.64},
{76.14,123.86},{47.79,83.40},{95.27,144.30},{44.13,98.20},
{19.97,37.36},{90.66,131.96},{75.41,117.80},{57.14,107.91},
{25.92,41.69},{90.86,130.36},{44.78,79.02},{23.00,29.10},
{91.67,118.13},{26.55,51.18},{41.60,74.91},{ 0.39, 6.79},
{86.31,102.08},{20.43,37.80},{ 5.39,28.65},{12.63,24.33},
{22.60,42.79},{ 1.77,14.54},{74.10,113.64},{54.46,87.67},
{18.64,49.32},{93.97,116.30},{42.62,87.04},{13.37,30.16},
{74.50,104.62},{18.28,67.85},{76.98,107.84},{25.89,57.35},
{13.52,42.87},{61.26,97.78},{ 5.97,31.34},{91.99,137.43},
{20.38,58.23},{ 9.59,31.56},{79.41,126.40},{89.90,134.36},
{73.18,111.44},{61.51,111.41},{99.96,147.82},{72.55,113.52},
{66.21,110.93},{36.47,59.41},{65.58,93.39},{24.93,51.71},
{58.00,95.89},{49.83,83.52},{53.35,89.98},{83.97,129.85},
{57.33,106.86},{53.94,98.13},{98.02,144.26},{47.28,72.52},
{45.48,100.70},{80.69,147.66},{96.14,140.01},{82.69,120.80},
{79.73,136.89},{11.42,27.51},{88.91,138.59},{25.53,51.26},
{ 2.49,37.14},{63.89,93.28},{90.96,138.02},{15.27,53.03},
{25.39,51.31},{31.77,55.54},{88.25,124.46},{67.66,108.26},
{90.23,112.02},{17.40,43.85},{78.38,137.07},{96.28,149.45},
{77.38,120.54},{56.49,107.27},{99.00,141.67},{36.35,58.18},
{97.41,132.64},{15.03,48.28},{42.48,81.20},{62.95,105.32},
{99.76,147.11},{85.18,140.95},{99.23,131.84},{21.09,44.44},
{45.12,75.22},{80.36,119.71},{61.37,84.74},{82.64,128.58},
{70.34,108.16},{83.63,116.26},{47.73,67.57},{17.56,48.42},
{23.26,42.12},{41.81,82.17},{18.48,33.63},{39.11,70.14},
{84.20,123.97},{67.20,113.97},{52.74,87.79},{81.66,131.54},
{45.90,93.69},{20.82,34.77},{86.35,122.38},{78.93,106.82},
{10.56,44.66},{51.20,104.61},{93.79,131.97},{15.71,43.06},
{99.16,156.47},{90.70,135.27},{41.85,77.91},{73.41,106.66},
{57.51,108.55},{53.06,115.27},{25.72,67.45},{ 8.03,27.74},
{57.91,101.56},{35.87,57.47},{98.33,145.81},{50.96,76.84},
{57.86,102.10},{17.21,44.21},{95.62,154.59},{76.92,114.77},
{25.32,60.66},{43.60,68.34},{42.68,73.98},{60.36,84.81},
{ 9.06,42.91},{ 4.16,18.44},{54.14,97.87},{ 4.87,35.92},
{75.38,112.62},{41.37,68.92},{88.16,163.96},{16.79,41.87},
{ 9.77,40.62},{69.66,125.12},{70.35,118.66},{71.99,97.87},
{63.66,111.29},{ 2.01,19.46},{64.63,122.89},{48.39,84.19},
{28.15,64.69},{46.17,83.91},{25.12,45.94},{82.23,118.70},
{57.69,95.98},{24.42,62.91},{15.81,35.58},{75.28,106.87},
{95.74,133.25},{67.78,107.42},{80.89,128.72},{10.39,38.37},
{15.31,35.73},{61.45,110.46},{11.15,44.99},{30.80,63.26},
{84.29,122.39},{29.17,47.34},{80.68,138.44},{81.17,117.86},
{ 8.47,32.78},{41.26,74.09},{43.50,71.18},{34.48,68.61},
{30.63,68.05},{88.63,137.28},{71.56,116.97},{21.03,39.12},
{88.20,116.24},{ 8.52,30.24},{95.79,137.27},{78.66,104.62},
{72.44,94.21},{71.60,106.34},{72.11,114.18},{34.50,59.18},
{22.85,60.95},{18.43,40.91},{69.24,119.69},{91.84,142.06},
{34.41,69.95},{95.06,136.92},{67.93,100.93},{46.96,71.82},
{63.92,102.14},{ 1.62,29.66},{95.24,133.60},{43.10,80.88},
{21.83,73.25},{35.01,62.42},{20.05,55.19},{18.64,45.92},
{40.28,75.26},{34.54,63.38},{84.74,117.68},{90.38,144.87},
{ 9.91,24.87},{62.97,102.14},{34.40,79.20},{67.34,89.48},
{48.53,85.13},{24.57,51.59},{81.95,117.78},{22.23,49.77},
{75.86,125.20},{60.45,99.78},{19.93,35.57},{48.62,78.46},
{88.49,120.71},{13.33,40.67},{52.03,93.38},{38.43,80.28},
{ 2.56,17.00},{18.39,58.10},{58.81,88.08},{75.76,96.69},
{69.78,98.83},{96.47,146.81},{47.32,79.89},{21.90,46.54},
{52.39,83.38},{75.49,107.96},{50.14,80.51},{41.54,73.80},
{76.07,117.48},{27.00,73.59},{81.59,122.88},{21.74,39.55},
{60.05,105.04},{75.68,102.72},{40.41,79.01},{ 0.32,24.82},
{50.06,106.14},{98.69,139.50},{64.17,109.26},{42.74,78.53},
{39.52,71.78},{55.14,97.37},{25.19,39.08},{99.31,142.63},
{67.50,91.86},{90.92,152.17},{81.99,129.38},{77.28,124.08},
{29.38,69.15},{ 3.81,41.93},{ 9.72,41.83},{25.75,53.09},
{57.28,85.11},{69.50,116.90},{20.00,51.46},{63.00,72.32},
{67.06,102.20},{37.85,64.86},{81.40,114.28},{13.32,58.41},
{67.21,103.77},{63.73,109.66},{91.43,141.66},{54.83,88.07},
{68.03,112.67},{ 0.51,27.76},{ 2.17,38.05},{36.26,66.58},
{72.67,116.52},{98.28,136.37},{85.27,128.64},{90.26,136.47},
{60.31,95.24},{32.77,58.94},{ 3.52,24.75},{15.98,45.49},
{94.25,145.90},{ 8.13,29.89},{61.13,81.38},{44.14,77.64},
{63.53,100.35},{49.35,97.92},{ 4.98,32.12},{25.53,57.45},
{ 8.63,41.62},{24.23,56.27},{93.30,137.92},{43.72,71.72},
{54.15,89.12},{ 3.42,36.34},{57.75,85.68},{51.90,87.74},
{85.14,137.82},{99.27,173.87},{82.53,124.94},{15.38,44.42},
{66.66,108.56},{64.12,99.41},{39.08,73.77},{25.42,58.25},
{ 1.29,36.39},{98.72,148.84},{70.09,112.06},{ 8.51,27.00},
{85.92,124.74},{88.32,127.04},{51.79,74.58},{36.46,62.45},
{49.29,85.33},{14.06,30.58},{24.83,34.82},{42.85,87.06},
{34.47,76.96},{59.16,90.44},{ 1.02,32.32},{61.80,108.22},
{72.52,95.83},{65.40,99.49},{53.32,93.79},{74.22,117.61},
{53.86,88.31},{39.84,80.11},{79.28,117.86},{34.57,76.73},
{21.69,55.55},{99.87,129.34},{72.12,108.86},{75.08,106.64},
{70.71,106.00},{18.35,67.45},{37.42,66.71},{ 0.70, 9.02},
{56.79,86.75},{74.04,100.45},{53.40,82.23},{42.13,70.45},
{82.43,123.55},{91.65,131.55},{94.99,153.70},{62.14,84.17},
{99.71,151.07},{33.24,73.77},{48.87,76.91},{68.57,118.95},
{14.28,46.22},{18.17,41.01},{95.93,133.32},{ 5.06,33.23},
{57.58,95.47},{18.71,39.10},{90.19,136.73},{26.98,50.08},
{11.36,26.14},{62.70,98.59},{49.32,80.54},{99.97,149.27},
{83.40,132.00},{25.30,48.62},{79.25,117.83},{81.09,109.23},
{31.46,51.02},{14.26,32.26},{33.53,52.63},{ 9.42,47.16},
{67.40,109.90},{18.56,32.79},{34.51,75.14},{49.00,77.38},
{15.69,50.80},{23.09,40.32},{32.03,67.86},{13.60,40.35},
{19.21,60.16},{78.56,111.57},{80.72,131.02},{50.19,79.64},
{55.60,81.78},{ 6.37,43.37},{42.78,74.85},{60.48,113.67},
{44.44,89.27},{54.02,90.24},{73.51,101.74},{16.41,56.73},
{70.94,104.90},{32.03,66.91},{13.12,49.71},{50.16,85.64},
{41.31,68.88},{69.25,123.25},{24.97,69.28},{40.80,86.30},
{32.28,67.01},{90.77,142.80},{66.77,104.70},{24.06,56.12},
{49.16,89.52},{46.10,95.56},{51.79,94.01},{56.11,100.66},
{88.49,126.71},{ 1.28,21.35},{35.55,64.10},{18.79,29.74},
{ 5.40,40.02},{92.32,129.89},{21.13,47.05},{ 5.14,32.16},
{60.89,104.41},{43.45,76.07},{98.91,160.53},{99.31,155.80},
{74.71,121.53},{62.33,98.98},{58.66,101.10},{51.51,93.03},
{51.69,90.42},{19.47,31.22},{85.75,108.87},{64.20,100.48},
{96.60,142.66},{67.99,102.48},{68.37,120.07},{29.81,44.77},
{96.55,142.74},{30.59,43.25},{73.94,108.44},{49.77,88.88},
{59.48,98.21},{41.21,61.86},{38.63,83.41},{86.98,140.40},
{93.34,134.69},{87.92,119.52},{40.93,61.87},{ 2.43,30.68},
{50.74,71.81},{37.13,52.43},{ 1.50,22.18},{99.06,143.48},
{ 1.67,27.67},{ 0.18,10.50},{54.13,77.05},{46.19,88.91},
{91.13,144.49},{ 8.95,28.33},{85.69,122.61},{50.30,95.60},
{48.63,103.49},{67.99,100.19},{69.21,112.13},{11.26,34.99},
{25.78,58.73},{84.35,112.36},{46.80,79.68},{69.54,117.99},
{40.30,74.33},{79.97,118.95},{23.28,55.71},{32.62,78.92},
{21.86,37.01},{ 5.07,22.57},{94.41,146.15},{40.14,60.81},
{95.80,125.35},{91.34,131.68},{72.55,113.56},{40.13,71.59},
{98.06,145.27},{90.55,144.08},{71.26,121.81},{33.85,71.13},
{85.74,142.63},{57.93,91.78},{ 7.63,39.30},{83.72,128.26},
{10.89,46.78},{39.79,66.98},{98.84,146.32},{84.62,123.91},
{23.16,31.94},{86.36,134.79},{44.19,63.74},{ 0.39,24.19},
{64.22,96.97},{66.47,103.78},{ 1.73,17.52},{22.25,36.77},
{31.88,59.39},{15.60,30.03},{16.08,41.91},{83.11,129.19},
{72.61,122.52},{19.02,41.06},{56.90,87.53},{65.85,97.02},
{81.40,120.35},{64.90,104.44},{73.35,119.00},{ 8.49,40.31},
{31.20,65.32},{28.29,75.05},{72.51,120.90},{20.42,48.84},
{71.46,111.59},{33.98,50.46},{72.48,111.29},{75.56,113.00},
{58.65,95.16},{23.66,44.95},{95.08,139.46},{80.12,115.20},
{67.77,101.97},{56.06,99.08},{99.03,138.47},{48.26,74.79},
{25.95,39.30},{85.20,137.70},{69.31,104.19},{86.19,122.91},
{37.99,87.47},{72.06,116.90},{ 5.66,28.92},{27.77,52.05},
{31.89,60.32},{18.01,48.92},{37.21,65.49},{73.76,107.20},
{ 0.32,-0.71},{93.75,133.48},{69.11,109.63},{11.01,55.84},
{43.48,73.99},{20.76,57.44},{75.50,105.00},{98.74,150.46},
{40.75,90.93},{61.67,103.30},{93.48,155.96},{35.52,61.62},
{32.30,78.52},{28.92,49.61},{60.97,87.11},{13.59,47.58},
{ 9.43,26.07},{58.00,107.90},{99.86,151.90},{34.01,57.82},
{39.02,59.14},{33.64,74.99},{ 2.28,20.21},{55.00,90.93},
{55.77,85.94},{79.17,134.03},{63.16,106.70},{17.58,32.28},
{24.29,34.68},{83.91,132.35},{96.44,129.86},{61.95,93.66},
{14.86,25.10},{15.53,33.29},{15.69,42.47},{80.60,126.11},
{16.01,46.33},{26.54,74.55},{ 2.67,37.10},{74.63,96.98},
{38.06,59.99},{56.59,96.87},{78.88,120.95},{87.56,121.75},
  {73.54,119.27},{16.84,44.09},{44.24,89.36},{76.02,123.64},
  {98.41,115.45},{12.11,48.19},{30.70,60.41},{55.51,100.49},
  { 0.26,37.11},{83.43,124.44},{49.92,111.30},{65.55,99.48},
  {77.61,119.44},{62.44,95.52},{21.80,61.06},{20.99,60.54},
  {93.10,129.45},{54.96,91.05},{10.22,48.48},{66.77,108.83},
  {40.83,87.14},{13.54,35.77},{31.44,62.92},{79.69,110.30},
  {67.07,100.59},{28.81,78.71},{52.95,97.30},{39.89,81.67},
  {58.79,75.89},{34.35,51.29},{38.03,64.97},{87.87,130.19},
  {39.73,52.43},{ 1.64,31.22},{91.15,147.58},{54.08,101.10},
  {53.53,74.54},{54.24,104.47},{15.04,51.28},{79.06,114.59},
  {93.83,138.37},{94.89,122.18},{52.63,86.22},{27.83,68.05},
  {54.51,94.07},{23.83,58.00},{86.88,141.66},{10.42,31.81},
  {55.43,84.31},{45.04,85.30},{95.69,121.78},{17.28,35.32},
  { 3.17,33.76},{51.61,69.81},{27.37,64.13},{88.92,160.98},
  {31.40,64.46},{33.35,59.91},{82.48,128.89},{50.46,98.13},
  {78.73,113.68},{70.08,115.27},{98.65,142.28},{ 9.15,50.95},
  {16.74,35.73},{32.92,72.02},{ 1.29,18.94},{75.79,123.45},
  {32.94,59.92},{61.72,81.50},{42.39,91.90},{70.15,108.81},
  { 2.90,29.10},{59.68,87.41},{69.85,108.66},{71.21,107.81},
  {24.09,46.47},{44.51,76.59},{ 7.30,34.83},{58.93,99.24},
  { 1.24,22.60},{84.27,132.21},{54.11,87.19},{39.18,75.93},
  {90.81,155.72},{67.68,88.19},{67.14,84.53},{53.98,86.47},
  {67.28,106.68},{ 8.49,36.74},{34.96,62.55},{59.01,82.94},
  {64.78,101.77},{66.24,110.82},{75.81,131.28},{62.82,76.02},
  {73.95,116.37},{20.40,38.76},{45.06,84.65},{47.64,82.81},
  {30.85,64.41},{77.10,112.67},{ 8.12,32.76},{39.56,53.41}
};
```

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



Simple Program				
Time elapsed was	82	ns or	0.000000082000	s
Time elapsed was	110	ns or	0.000000110000	s
Time elapsed was	61	ns or	0.000000061000	s
Time elapsed was	68	ns or	0.000000068000	s
Time elapsed was	83	ns or	0.000000083000	s
Time elapsed was	81	ns or	0.000000081000	s
Time elapsed was	77	ns or	0.000000077000	s
Time elapsed was	57	ns or	0.000000057000	s
Time elapsed was	67	ns or	0.000000067000	s
Time elapsed was	61	ns or	0.000000061000	s
total	747	ns or	0.000000747000	s
Mean	74.7	ns or	0.000000074700	s

Figure 15 running program for 10 time of liner regression



Multithread Program				
Time elapsed was	236663119	ns or	0.236663	s
Time elapsed was	235903601	ns or	0.235904	s
Time elapsed was	235863296	ns or	0.235863	s
Time elapsed was	238378063	ns or	0.238378	s
Time elapsed was	238378064	ns or	0.237744	s
Time elapsed was	238378065	ns or	0.237711	s
Time elapsed was	238378066	ns or	0.23731	s
Time elapsed was	238378067	ns or	0.234256	s
Time elapsed was	238378068	ns or	0.23716	s
Time elapsed was	238378069	ns or	0.235251	s
Total	2377076478	ns or	2.36624	s
mean	237707647.8	ns or	0.236624	s

Figure 16 Running multithreading program for 10 time of liner regression

Write a short analysis of the results.

2 CUDA

2.1 Password Cracking

```
Paste your source code for your CUDA based password cracker here
```

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

Write a short analysis of the results

2.2 Image Processing

```
Paste your source code for your CUDA based image processing.
```

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

Write a short analysis of the results

2.3 Linear Regression

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

```
Paste your source code for your CUDA based linear regression
```

Write a short analysis of the results

3 MPI

3.1 Password Cracking

```
Paste your source code for your MPI based password cracker here
```

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

Write a short analysis of the results

3.2 Image Processing

```
Paste your source code for your MPI based image processor
```

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

Write a short analysis of the results

3.3 Linear Regression

```
Paste your source code for your MPI based linear regression
```

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

Write a short analysis of the results

4 Verbose Repository Log

```
Paste your verbose format repository log here. With subversion this can be achieved by the following:

        svn update

        svn -v log > log.txt

        gedit log.txt

Then select, copy and paste the text here
```