6CS005 - High Performance Computing

Student-Id: 1929076

Student name: Sanjay Tamang

B.Sc. (Hons) Computer Science

Submitted date: 2019/11/30

Submitted to: Dr Kevan Buckley

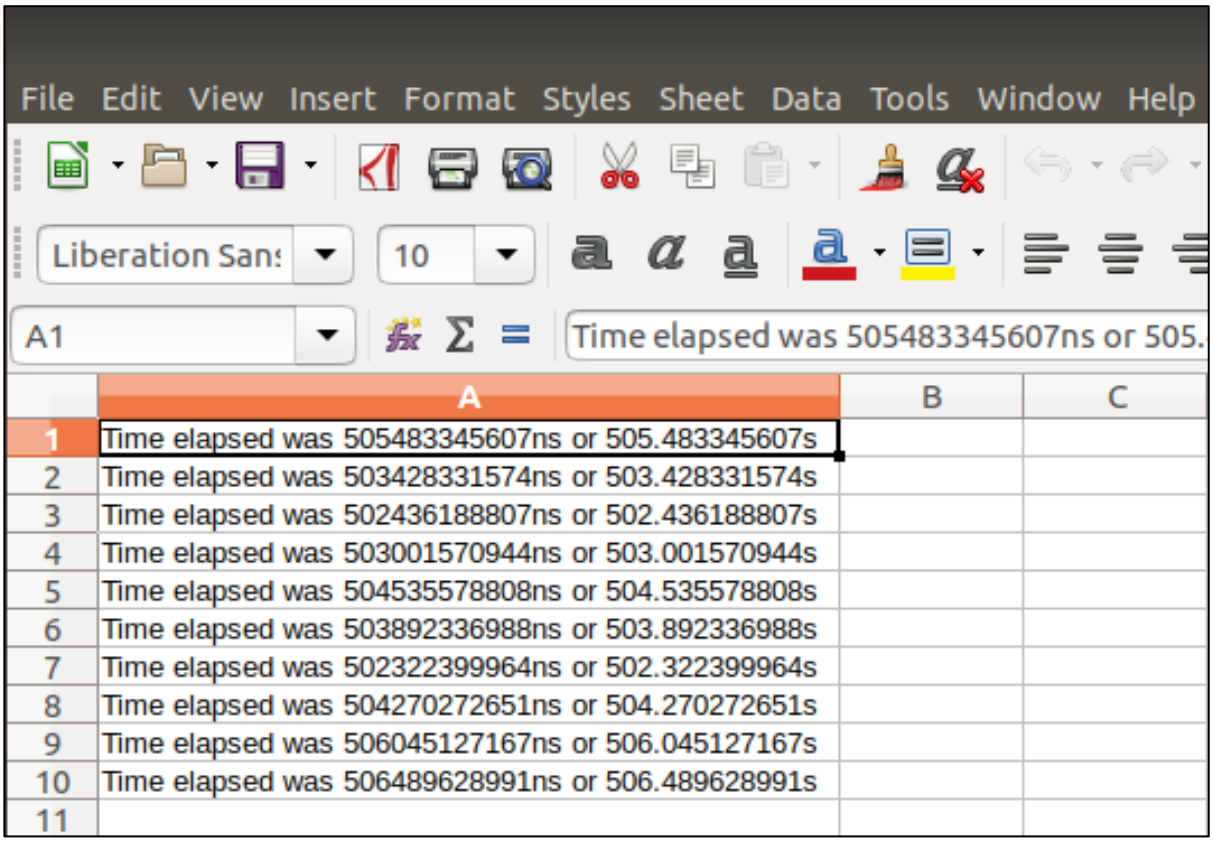
Table of Contents

1 POSIX Threads

1.1 Password Cracking

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

In i3 processor 7th generation:

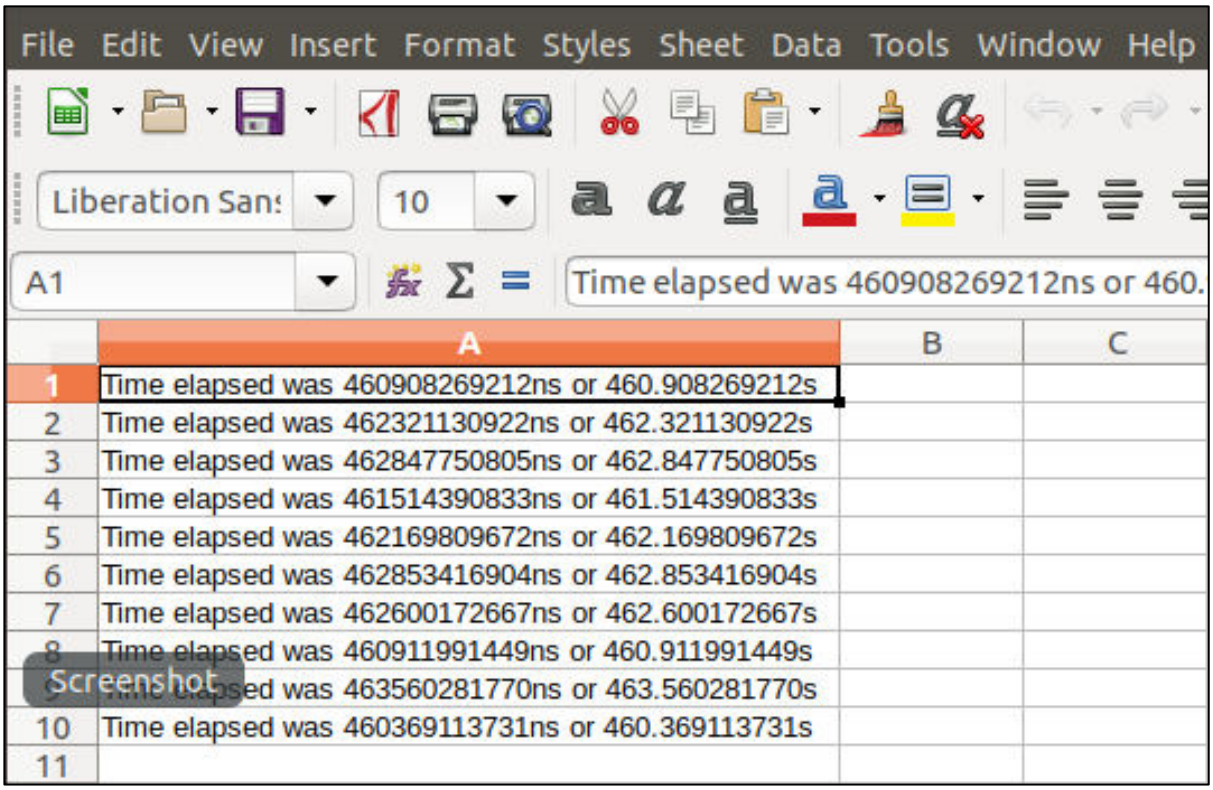


In i5 processor 9th generation:


Open
two_initial.txt
~/Downloads/passwordcrack_withoutThread
Save
CrackAZ99_twoInitial.c
two_initial.txt
14844 FS43 6KB$Qbbkv0yc2gl4F6P94SLCZHSA1J7/ow2ueJ/.9zNo5pCcln3t/
OIp.zBYhtE.AWrgiYfylIiHdwrJqT2BAlxpg1
14845 FS44 6KB$pty58/fNwhYkxl48GyQl14.SWdWjQxm9U2XZ2oW./
p9LwRmfcyNTVya38J4jo1zZJX9WNSmuTD8GMIaodizbQ.
14846 FS45
6KB$q5dR1CmRo3v5L7R.x122ubK3SDFpOKI280vUugL3PnwoCCWpRS2p22ucINxsLMsR8x8PE5A0.j0l75puY8rFR/
14847 FS46 6KB$r.uTMO.kblb3z6ZlrxXILdpTqSmmXk2sOHdskjn64.nUemNXX2bBawKjT3IpazKuY4k5m33hc86q3Y1SqD/
d0.
14848 FS47 6KB$jyDvGJlpBoZ7V0LmBQMe8IRWBBOs5iptBLdOhT4LNJClRiXwfx4ul/
IlCXEgzYOUjIhmBUJKNfHPVmJP3dueR1
14849 FS48
6KB$HpoM5MR80vHEOYsSHTPkprX4LXKX2MDIMkUh4eLT7yhDWGUC27P80rWZY.enh.pCg736XNtmhmwExOAxVTkdf.
14850 FS49 6KB$Wk/T3WwO0ec0ebXkcdrmgfyIWEBbxdsReORrX.yRu.eBOeZfdankIoNS5PMs.X/
QRhmyKLFp0rAkh.gUZnnQu.


Open
two_initial.txt
~/Downloads/passwordcrack_withoutThread
Save
CrackAZ99_twoInitial.c
two_initial.txt
20145 HT44 6KB$uOMAgfvoDMUrxhHWBmY2TO2uWWxCzQIQHS9yRLfmo2aMB9l9nBrNrFYjBnkaYi.gYvplYSVzQ235Ya61b9Y/
P.
20146 HT45 6KB$Ekez0GXP05p994IvJBOw8u7Z.TEGunznl8Nc06mSxzT0/3Lq/
UmBz0ROSoWq8.7A9aBnQWjBalk677EWRsdvF.
20147 HT46 6KB$5MUM8ErA2CYdGYk.ZtsPWvobaob8xJQiuFf/
s13adux8ZuVu99XHPlTiwO9QYZp.W4qxhPKyLwLjQcBMf6B8Q1
20148 HT47
6KB$n1B.Bf8mckpiSJv8ZPLNsLDv75zffZL5PecbjAmRuAj.lynQS9QUU0QpQ69w4SaC5b2jR0FxE.UyB9Sud7olv0
20149 HT48 6KB$iyAdOw/
ziDVBE0sXz8H3YRvGMVpqgV0DTg0dVbtPUyheOGYQGWP0C0g4hXnGTMZtUT0NXtmeaMY1Q6ykJqcTw0
20150 HT49 6KB$qeKfdGL7z9pxjrE5dt4cm9ZwE1azDXMUp/
mb7NIKtItj2Z6H0OUnPArTGEkd2fTBiucn7QOzUX2aPOGzhFIJB1
20151 HT50 6KB$6TolE70r9vARxQI.L7fce1rqe/


Open
two_initial.txt
~/Downloads/passwordcrack_withoutThread
Save
CrackAZ99_twoInitial.c
two_initial.txt
44286 RA85 6KB$KyOsM0X7KtljF.FKO4KcCNVliAm0Of6kwuR4G6EZixTL3mWdCKPNnzFW/n7UYawDOcvs2j0pGOQNF/
UsYRsy61
44287 RA86
6KB$f7G8Qck0v4qJsQUgzRJ0bdbpQoVPWxRqIsbAkckZXlamiX5A2PhvM3VkHS5DRxFu2HHZpqhuWlepNgrEG1m3l0
44288 RA87 6KB$NPSR22FtPf31BB1FJJbswQpGZjIIreBag379t79LMkFVgsLJxFzuC4R5IphlCJ6j9ADkOxiIg6uzS/
tnO.Byd/
44289 RA88 6KB$UYmzOVDyHlb3mrpUYg3USD.dKf0hEmwQ/
oe6KohN3OC9a1XmjJ8DQQVhjKZjdGdooVmeewF0O2VZbVdtJ3n8k/
44290 RA89 6KB$Uz4cD9uzcYjtg9/
zNnA4wdLtqlTWw42taHPdqzfJYQOmv2Ct79UJ8e11XtqdxzH3E58trHonpZFDOwYRwJPGs1
44291 RA90 6KB$4Ae.WPLvXf6vPlOkTK69SpapmZ.DkyU2QnCKid4ugODGU5U9SYCGh4a9IclOxCPkqDOwPCU/
gRdyxoZ5N18nU/
44292 RA91 6KB$FG1.3zPDYn0T4pp9MmN7Qr2lvCA2.13EjkMY86pWvh.AkAKpuJo0ZEZHUhlB7JOMJP3K9cXl/
AMpaeBCXQLBc/



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.

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

/**********************************************************************
*******
  Demonstrates how to crack an encrypted password using a simple
  "brute force" algorithm. Works on passwords that consist only of 2
uppercase
  letters and a 2 digit integer. Your personalised data set is included
in the
  code.

  Compile with:
    cc -o CrackAZ99_twoInitial CrackAZ99_twoInitial.c -lcrypt

  If you want to analyse the results then use the redirection operator
to send
  output to a file that you can view using an editor or the less
utility:

    ./CrackAZ99_twoInitial > results.txt
```

```
 Dr Kevan Buckley, University of Wolverhampton, 2018
*****************************************************************************
******/
int n_passwords = 4;

char *encrypted_passwords[] = {

"$6$KB$3MiAO5oLs/.coZCPQ2QYOy8Ozo3v7QzGdwBEv3N7E0pJen3CJ63DmYXIZz6KEsykHmGsu3Dh
1KCNe0niN0wvx/",


"$6$KB$jyDvGJlpBoZ7V0LmBQMe8IRWBBOs5iptBLdOhT4LNJClRiXwfx4ul/llCXEgzYOUjIhmBUJKNfHP
VmJP3dueR1",


"$6$KB$iyAdOw/ziDVBE0sXz8H3YRvGMVpqgV0DTg0dVbtPUyheOGYQGWP0C0g4hXnGTMZtUT0NXt
meaMY1Q6ykJqcTw0",


"$6$KB$Uz4cD9uzcYjtg9/zNnA4wdLtqlTWw42taHPdqzfJYQOmv2Ct79UJ8e11XtqdxzH3E58trHonpZF
DOwYRwJPGs1",
};

/**
 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 crack(char *salt_and_encrypted){
  int s, a, n;    // 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 encryprivate static int MAX = 5;
```

```
/// <summary>
/// Select this entire code block and
/// click to the "Format text as code"
/// to change the styling. We will use your style
/// preference when formatting.
/// </summary>
public void TestMethod()
{
  // Try highlighting the line below.
  // We will try to highlight using your highlighting preference.
  Console.WriteLine("Select this line and click Highlight line button");
}pted password
 int count = 0;   // The number of combinations explored so far

 substr(salt, salt_and_encrypted, 0, 6);

 for(s='A'; s<='Z'; s++){
  for(a='A'; a<='Z'; a++){
   for(n=0; n<=99; n++){
    sprintf(plain, "%c%c%02d", s, a, n);
    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++)
        {
          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;
}
```

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

In i3 processor 7th generation:

In i5 processor 9th generation:

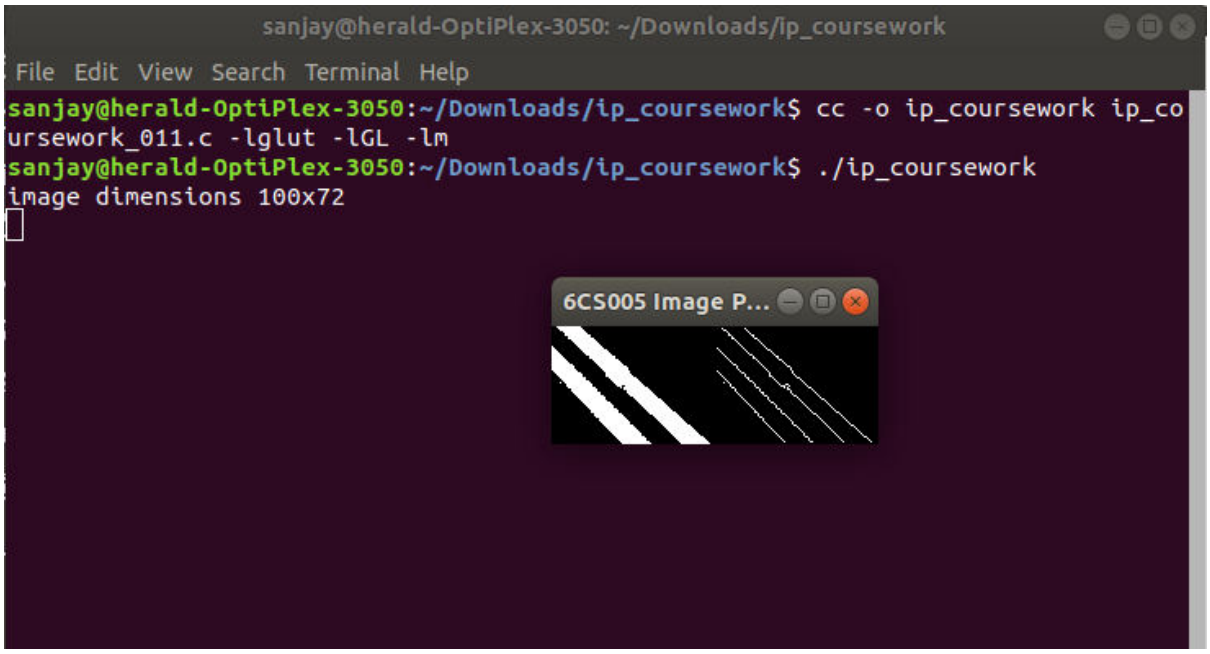
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <crypt.h>
#include <time.h>

/**********************************************************************
*******
  Demonstrates how to crack an encrypted password using a simple
  "brute force" algorithm. Works on passwords that consist only of 2
uppercase
  letters and a 2 digit integer. Your personalised data set is included
in the
  code.

  Compile with:
    cc -o CrackAZ99_threeInitial CrackAZ99_threeInitial.c -lcrypt

  If you want to analyse the results then use the redirection operator
to send
  output to a file that you can view using an editor or the less
utility:

    ./CrackAZ99_threeInitial > results.txt

  Dr Kevan Buckley, University of Wolverhampton, 2018
**********************************************************************
******/
int n_passwords = 4;

char *encrypted_passwords[] = {

"$6$KB$u3Udg2.tSd3sUGHvTvL8U7K6qdVUrNwLOo4yzgerr0ty32wmymZ0x23k.6btiED1H8/qqajavj1
pUwYNyLcmf0",


"$6$KB$Q2EIAiLddE.KV3lIzGOsjrzj4CcowgDU3RK4lEbs7WSPms86jhIf8KzyOIJiApCKg13h6p3z06d3646
ER0pVf/",


"$6$KB$e5KEpWhudmb6hfQLFwyGTfeauwNCIOetz/G1n65uEwAS5daJMvqIMt77.y.kWnuAefEqTMkA
SsPCnMcIGlFzy.",


"$6$KB$zxzPi8wWZCuIi25BYR7EFgdD4WDOcGXzlQjNKrIRftsWBhEcTiCBB.7f48YWLETLHha/qBYjzWV
GuwJI7xRgG1",
};

/**
 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 crack(char *salt_and_encrypted){
  int s, a, n, j;    // 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(s='A'; s<='Z'; s++){
    for(a='A'; a<='Z'; a++){
      for(n='A'; n<='Z'; n++){
        for(j=0; j<=99; j++){
        sprintf(plain, "%c%c%c%02d", s, a, n, j);
        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++)
        {
          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;
}
```

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>

/*******************************************************************************
 Displays two grey scale images. On the left is an image that has come from an
 image processing pipeline, just after colour thresholding. On the right is
 the result of applying an edge detection convolution operator to the left
 image. This program performs that convolution.

 Things to note:
  - A single unsigned char stores a pixel intensity value. 0 is black, 256 is
    white.
  - The colour mode used is GL_LUMINANCE. This uses a single number to
    represent a pixel's intensity. In this case we want 256 shades of grey,
    which is best stored in eight bits, so GL_UNSIGNED_BYTE is specified as
    the pixel data type.

 To compile adapt the code below wo match your filenames:
  cc -o ip_coursework ip_coursework_011.c -lglut -lGL -lm

 Dr Kevan Buckley, University of Wolverhampton, 2018
*******************************************************************************/
```

```
#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 main(int argc, char **argv) {
 signal(SIGINT, sigint_callback);

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

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

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

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

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

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

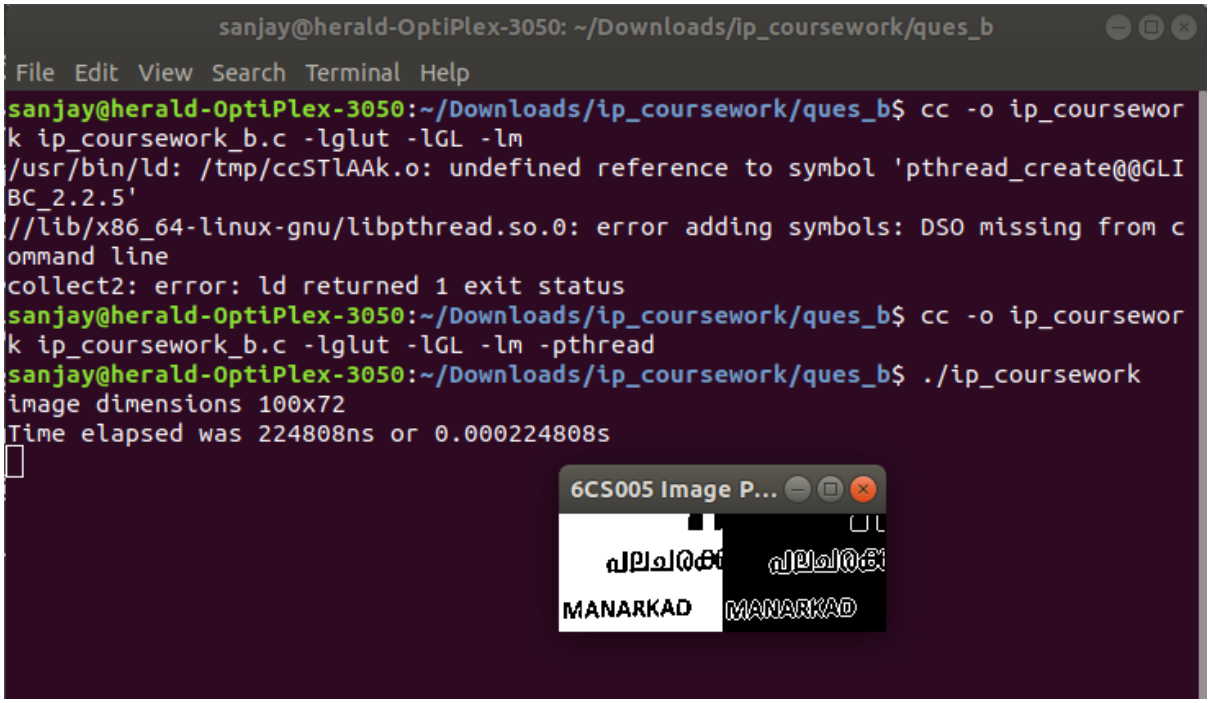
255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,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

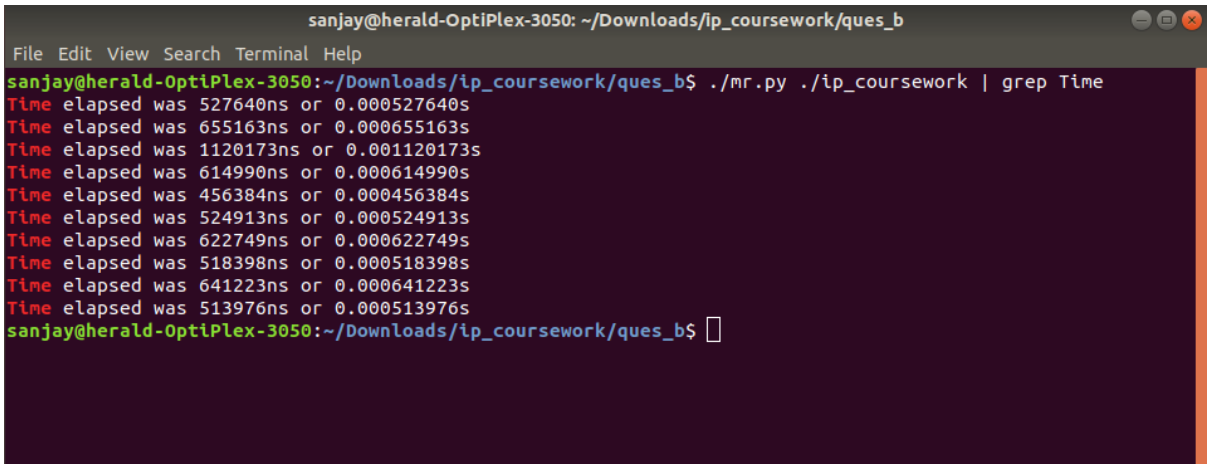
0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,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

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

```
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
 255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
};
```

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.





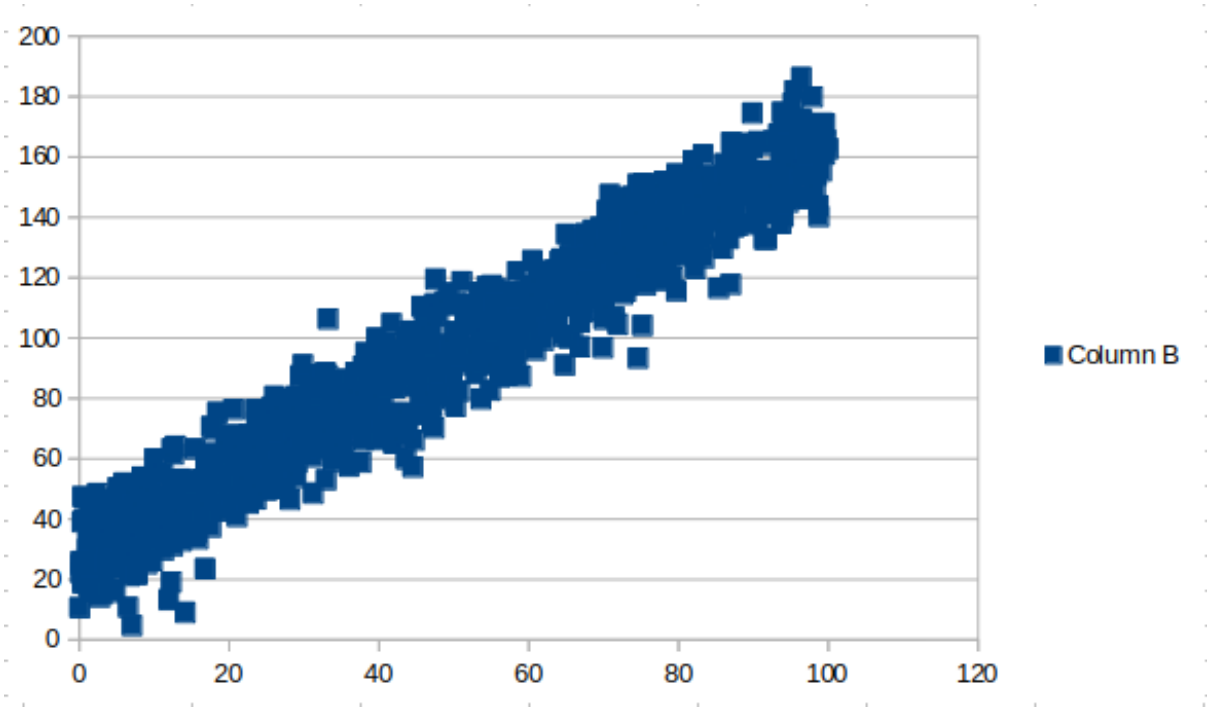
Insert an explanation of the results presented in the above table.

C13

	A	B	C	D	E	F
1	Time elapsed was 182874ns or 0.000182874s					
2	Time elapsed was 257318ns or 0.000257318s					
3	Time elapsed was 421447ns or 0.000421447s					
4	Time elapsed was 218200ns or 0.000218200s					
5	Time elapsed was 226635ns or 0.000226635s					
6	Time elapsed was 666919ns or 0.000666919s					
7	Time elapsed was 818187ns or 0.000818187s					
8	Time elapsed was 526428ns or 0.000526428s					
9	Time elapsed was 690399ns or 0.000690399s					
10	Time elapsed was 539112ns or 0.000539112s					
11						

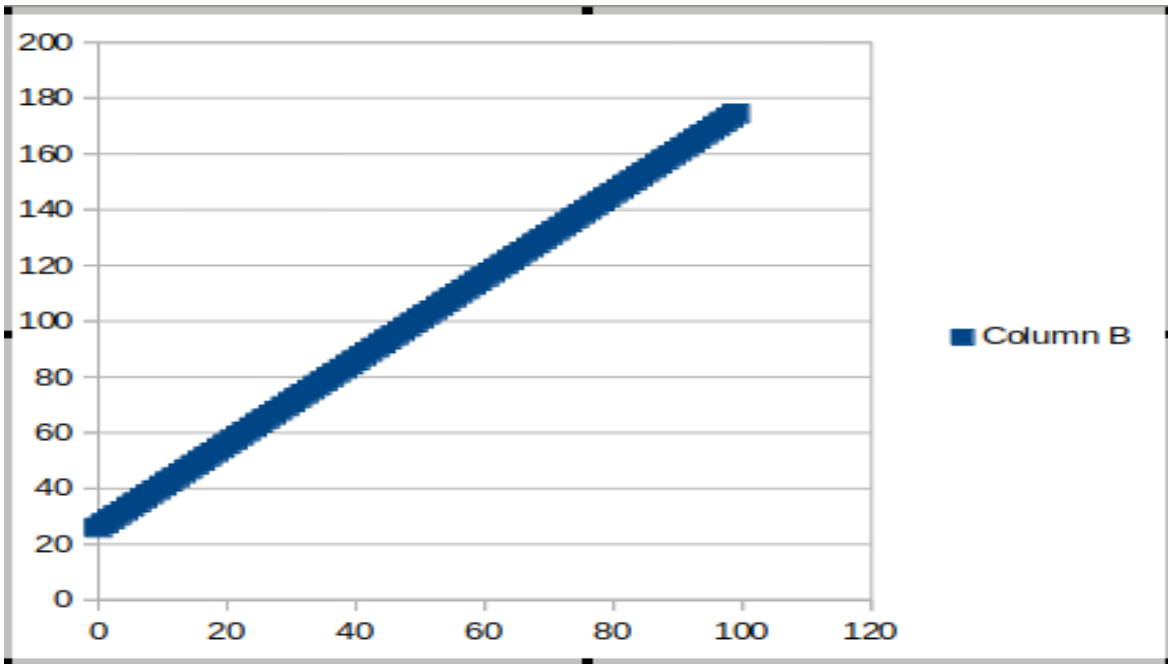
1.3 Linear Regression

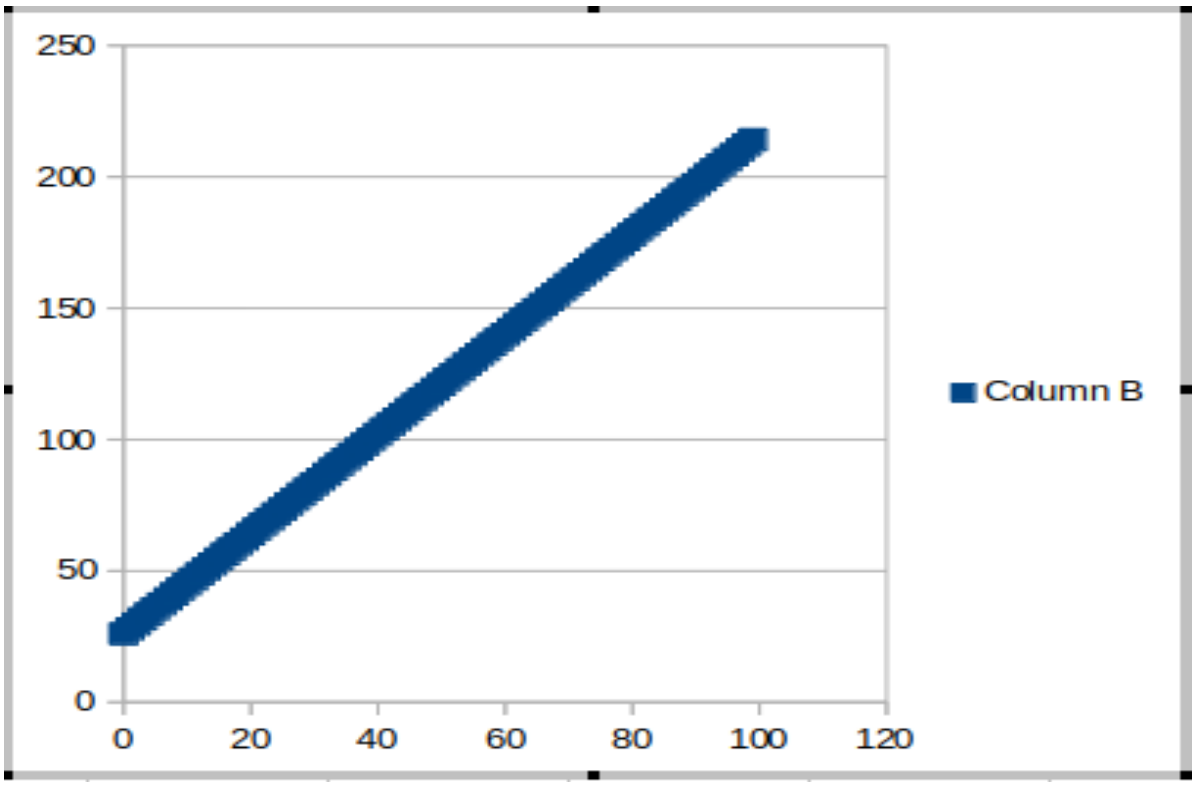
Insert a scatter plot of your data.

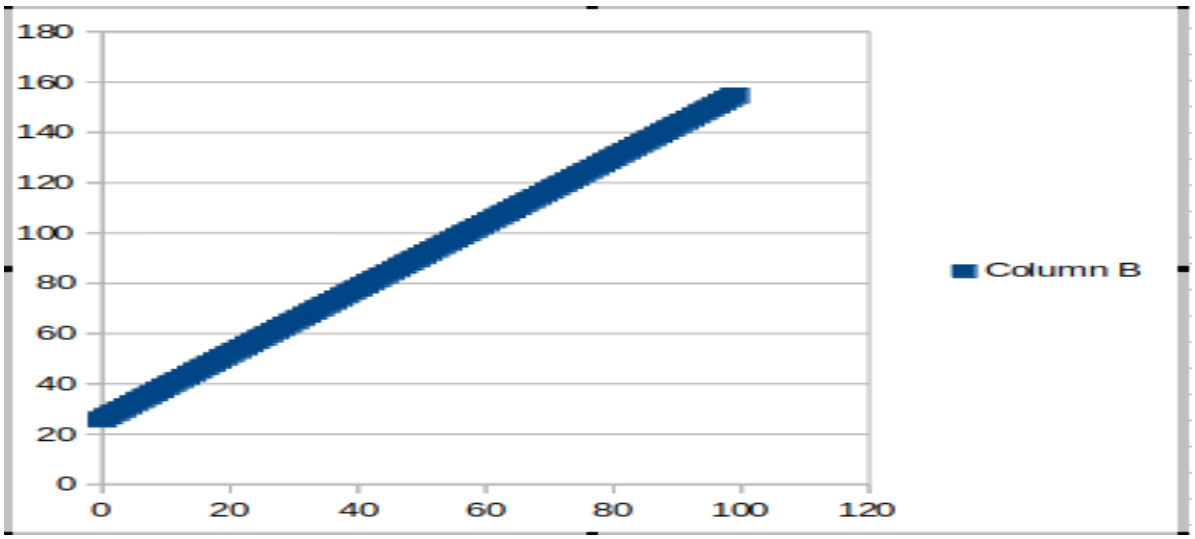



```
best m,c is 1.400000,25.870000 with error 10.119989 in direction 0
best m,c is 1.400000,25.880000 with error 10.119956 in direction 0
best m,c is 1.400000,25.890000 with error 10.119934 in direction 0
best m,c is 1.400000,25.900000 with error 10.119922 in direction 0
best m,c is 1.400000,25.910000 with error 10.119919 in direction 0
best m,c is 1.400000,25.920000 with error 10.119919 in direction 0
minimum m,c is 1.400000,25.910000 with error 10.119919
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression$
```

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








```
sanjay@herald-OptiPlex-3050: ~/Downloads/linear_regression/linear_regression_2
File  Edit  View  Search  Terminal  Help
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression/linear_regression_2$  cc -o lr01 lr01.c
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression/linear_regression_2$ ./lr01 1.5 25.95 > lr01_r
esults.csv
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression/linear_regression_2$ ./lr01 1.9 25.99 > lr01_r
esults_2.csv
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression/linear_regression_2$ ./lr01 1.3 25.89 > lr01_r
esults_3.csv
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression/linear_regression_2$
```

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

/*******************************************************************************
 * This program takes an initial estimate of m and c and finds the associated
 * rms error. It is then as a base to generate and evaluate 8 new estimates,
 * which are steps in different directions in m-c space. The best estimate is
 * then used as the base for another iteration of "generate and evaluate". This
 * continues until none of the new estimates are better than the base. This is
 * a gradient search for a minimum in mc-space.
 *
 * To compile:
 *   cc -o linear_regression linear_regression_2.c -lm
 *
 * To run:
 *   ./linear_regression
 *
 * Dr Kevan Buckley, University of Wolverhampton, 2018
 *******************************************************************************/
```

```
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, docc -o linear_regression linear_regression_2.c -lmuble 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[] = {
  {77.58,134.54},{79.03,131.77},{72.93,146.29},{65.61,133.25},
  {77.28,135.25},{77.92,118.99},{69.04,129.61},{68.76,116.10},
  {83.24,133.43},{65.85,100.77},{68.15,128.88},{73.52,128.43},
  {81.48,131.82},{85.82,147.98},{69.10,135.29},{ 7.88,45.72},
  {53.84,101.83},{18.31,44.95},{75.78,138.12},{73.99,128.86},
  {41.99,66.32},{81.52,129.96},{58.97,107.13},{51.49,108.22},
  {98.76,140.25},{73.85,146.91},{ 0.05,10.60},{19.33,48.49},
  {97.88,171.02},{56.93,99.12},{31.03,60.59},{57.63,110.20},
  {11.87,47.48},{ 7.80,50.44},{53.43,115.13},{19.55,65.05},
  {68.84,120.68},{ 3.97,15.72},{72.57,130.78},{89.28,151.71},
  {37.92,90.62},{15.68,35.51},{37.12,73.12},{77.72,123.27},
  {10.05,59.94},{72.73,124.45},{54.78,106.88},{17.23,52.72},
  {56.87,105.58},{74.69,120.59},{32.33,74.49},{76.72,132.91},
  {63.09,115.96},{23.19,65.78},{98.43,154.03},{39.68,99.96},
  {49.02,84.02},{18.92,42.64},{45.15,86.54},{74.60,93.30},
  {78.75,138.28},{72.78,118.03},{71.69,141.68},{92.14,164.07},
  {67.49,113.89},{47.48,111.31},{76.04,133.42},{81.33,127.26},
  {47.56,119.47},{ 4.54,41.48},{16.05,52.62},{21.22,51.33},
  {48.99,97.30},{69.92,96.79},{63.05,114.50},{41.22,98.30},
  {33.22,106.31},{41.87,72.13},{32.82,77.58},{41.42,74.83},
  {64.30,118.39},{93.41,167.69},{32.95,78.41},{90.81,138.22},
  { 3.10,20.24},{37.18,82.94},{ 3.65,30.48},{71.85,136.47},
  {68.50,125.08},{51.77,108.22},{64.46,110.70},{88.37,150.71},
  {84.28,138.95},{68.61,135.73},{ 9.52,36.37},{35.29,68.37},
  {50.33,111.40},{44.68,84.04},{30.88,61.56},{78.06,140.99},
  {96.67,159.23},{35.06,67.89},{94.03,154.50},{47.05,77.89},
```

{77.21,142.06},{71.44,106.89},{46.00,86.88},{43.02,94.70},
{96.39,172.58},{84.60,145.66},{57.94,97.23},{47.44,107.40},
{68.50,127.39},{86.16,157.69},{80.83,153.81},{86.05,147.56},
{61.56,114.61},{ 8.31,26.65},{66.85,97.08},{45.73,89.44},
{33.46,75.74},{30.37,65.41},{ 1.41,40.84},{57.21,110.64},
{ 9.03,25.18},{31.07,71.68},{65.36,99.96},{85.54,144.59},
{ 5.27,25.71},{85.81,153.14},{50.95,100.15},{72.98,115.53},
{45.48,84.24},{47.33,92.49},{32.24,75.53},{46.51,92.66},
{ 1.13,32.34},{75.20,104.12},{97.30,162.16},{64.64,125.57},
{80.38,129.98},{97.47,158.25},{16.76,47.51},{60.21,117.34},
{83.28,160.77},{84.76,151.90},{62.84,118.24},{73.64,119.24},
{67.50,113.77},{61.53,109.68},{38.88,70.80},{19.85,66.41},
{ 9.82,49.50},{10.77,38.24},{58.48,100.30},{73.04,131.43},
{72.94,123.65},{78.57,138.70},{70.02,127.93},{39.55,76.36},
{96.37,165.84},{96.00,170.77},{32.88,88.54},{69.63,132.37},
{55.74,116.24},{93.21,143.10},{93.40,167.42},{92.71,165.31},
{42.14,83.44},{82.28,123.03},{85.36,116.48},{17.61,54.92},
{39.26,90.51},{24.90,51.95},{30.49,87.85},{64.59,119.06},
{86.99,117.78},{63.80,103.91},{ 8.34,53.95},{74.15,120.72},
{25.94,57.77},{39.42,81.83},{24.37,62.28},{69.78,124.58},
{99.34,165.09},{55.66,111.58},{27.39,64.28},{70.79,124.03},
{41.47,94.31},{ 8.71,34.58},{72.63,124.81},{47.26,82.74},
{26.27,50.15},{ 1.52,25.49},{44.00,98.71},{55.50,106.20},
{20.74,45.68},{13.32,34.67},{81.66,144.46},{81.91,158.70},
{81.26,129.58},{ 3.07,26.91},{52.99,88.14},{31.39,71.70},
{93.33,144.37},{97.05,146.19},{29.52,87.47},{87.38,136.79},
{32.41,66.99},{26.04,71.12},{ 9.53,53.59},{ 1.11,29.88},
{12.79,46.21},{15.46,35.89},{58.97,107.21},{23.82,63.65},
{29.67,79.03},{70.25,134.07},{84.59,139.57},{31.29,63.46},
{24.91,51.96},{61.28,106.85},{86.41,135.44},{80.47,141.58},
{23.98,66.50},{26.80,71.74},{ 3.51,23.79},{86.54,133.49},
{93.72,150.37},{29.54,65.96},{22.70,55.04},{66.50,111.49},
{89.59,137.73},{55.73,102.20},{97.86,169.04},{70.47,126.63},
{29.88,91.13},{88.32,153.41},{18.44,75.32},{52.53,111.27},
{42.68,89.48},{74.59,150.94},{25.61,71.65},{61.35,120.49},
{79.88,147.72},{34.93,84.92},{89.31,164.47},{81.10,151.61},
{43.20,95.75},{83.73,143.72},{78.74,121.67},{29.09,58.87},
{97.26,170.20},{38.28,87.00},{25.56,67.05},{12.24,45.69},
{55.06,106.98},{48.80,110.95},{11.17,33.96},{44.58,86.15},
{81.70,135.88},{65.19,114.68},{51.50,91.24},{ 9.79,26.25},
{11.27,29.73},{40.51,91.85},{ 8.49,38.65},{36.87,88.64},
{63.71,113.51},{27.58,52.49},{61.46,104.57},{15.75,41.36},
{53.24,109.89},{99.34,171.13},{75.16,141.62},{52.48,104.49},
{77.50,130.99},{22.34,61.00},{97.91,179.94},{45.72,94.47},
{68.56,123.58},{51.80,106.92},{27.50,70.84},{99.70,165.48},
{41.54,73.96},{22.66,52.14},{77.55,141.12},{46.21,103.52},
{67.21,111.92},{93.99,140.47},{57.29,112.62},{97.03,160.38},
{51.70,109.48},{25.76,77.13},{57.51,102.99},{99.54,170.94},
{14.17,36.47},{31.38,73.72},{20.97,55.18},{98.22,154.72},
{75.90,150.32},{76.89,128.23},{87.86,137.32},{95.30,177.80},
{75.05,136.60},{66.05,119.53},{95.21,155.75},{21.06,40.86},

{ 4.66,23.21},{80.09,133.89},{53.08,96.42},{90.40,144.09},
{42.33,91.78},{38.22,88.40},{13.64,46.53},{56.14,87.09},
{87.52,146.52},{25.07,56.90},{42.16,89.42},{58.45,107.26},
{88.53,147.96},{11.94,13.24},{31.36,72.36},{22.60,45.26},
{14.14, 9.04},{31.07,65.58},{10.31,36.34},{79.58,142.26},
{71.32,124.24},{48.17,90.68},{70.85,147.53},{ 1.06,22.34},
{44.73,66.17},{ 3.82,44.30},{43.63,68.95},{77.07,141.84},
{19.82,60.62},{81.78,133.03},{74.37,127.74},{22.86,61.66},
{73.16,133.98},{ 5.16,50.36},{98.42,162.94},{ 6.70,21.25},
{44.53,57.25},{31.24,48.57},{58.95,87.36},{20.63,76.35},
{24.94,49.55},{84.49,142.92},{67.44,109.26},{46.62,82.23},
{23.44,54.19},{61.94,109.84},{46.07,109.41},{12.37,63.27},
{79.74,154.32},{59.51,115.51},{29.65,73.96},{28.95,53.76},
{16.23,42.07},{53.04,93.01},{70.45,142.17},{42.32,90.28},
{ 7.52,38.90},{89.11,156.32},{55.85,99.34},{43.11,73.82},
{87.99,161.98},{62.07,122.27},{37.70,78.51},{16.80,23.52},
{17.99,46.90},{48.52,97.82},{71.90,104.53},{ 5.62,31.03},
{33.70,72.22},{47.80,95.45},{ 2.35,23.75},{99.11,155.44},
{30.41,64.35},{77.56,120.38},{20.67,54.54},{61.00,109.95},
{91.42,132.66},{35.90,71.42},{10.30,44.65},{66.58,115.25},
{15.09,37.91},{ 0.51,26.02},{87.59,141.62},{73.09,139.44},
{94.66,145.08},{ 8.98,37.67},{28.03,67.82},{46.73,99.23},
{83.08,152.50},{78.90,134.38},{39.77,82.02},{15.50,40.58},
{28.11,56.13},{74.68,135.56},{11.16,55.02},{ 5.14,33.78},
{49.08,97.02},{92.22,141.82},{44.62,74.64},{ 4.87,46.68},
{26.94,54.94},{93.69,137.95},{27.15,68.43},{58.18,92.82},
{ 9.11,39.57},{60.72,109.23},{23.55,76.29},{32.96,52.96},
{54.94,107.89},{84.26,139.17},{ 5.88,44.96},{34.11,84.64},
{79.62,137.14},{82.69,135.97},{ 0.66,39.87},{13.50,53.25},
{78.16,137.54},{70.09,127.51},{85.92,129.78},{41.57,95.04},
{39.71,66.31},{76.07,130.41},{86.71,133.07},{65.12,111.93},
{12.97,51.01},{65.16,116.38},{56.48,98.70},{ 9.80,42.81},
{ 5.18,35.85},{95.53,181.91},{ 2.92,34.44},{36.81,74.04},
{ 2.13,43.15},{91.28,149.80},{87.77,141.32},{93.66,157.80},
{49.79,79.73},{39.20,85.13},{54.77,100.16},{41.20,88.31},
{92.11,155.37},{25.32,50.02},{28.12,46.50},{67.93,123.95},
{83.51,134.50},{37.63,58.85},{30.86,79.65},{72.77,116.35},
{91.16,145.60},{73.88,136.80},{23.10,49.08},{33.67,63.70},
{15.61,40.30},{76.42,143.52},{68.42,129.00},{74.41,122.91},
{84.68,148.78},{43.79,73.53},{71.62,117.52},{53.51,99.79},
{75.75,145.10},{ 7.75,45.26},{75.86,125.12},{ 5.29,31.29},
{ 6.69,37.78},{38.95,85.00},{34.78,63.05},{54.43,116.98},
{ 4.73,45.92},{89.84,174.55},{73.11,143.57},{56.94,99.64},
{36.01,57.50},{86.39,148.41},{13.63,32.81},{77.70,150.75},
{89.05,161.83},{12.28,19.16},{32.76,88.11},{36.49,72.23},
{ 1.14,18.31},{41.95,65.08},{83.81,149.37},{21.24,61.50},
{80.01,133.29},{ 6.53,10.74},{60.68,114.32},{ 2.43,42.40},
{22.99,61.42},{34.94,78.64},{57.13,87.55},{55.04,100.25},
{48.48,91.98},{58.36,93.98},{96.53,172.02},{44.57,94.54},
{15.19,52.72},{95.42,172.51},{30.72,69.65},{72.21,145.30},
{ 2.72,37.30},{ 6.19,35.84},{90.67,165.08},{ 6.21,30.23},

{59.80,108.35},{30.66,73.13},{38.36,80.84},{64.82,91.04},
{35.15,64.36},{60.72,108.89},{41.00,92.74},{23.08,48.35},
{47.07,72.00},{12.37,49.11},{67.90,120.63},{78.01,129.50},
{91.24,155.48},{30.14,65.75},{35.32,61.62},{18.09,69.49},
{64.99,119.13},{ 7.32,44.73},{ 0.36,39.11},{37.08,81.71},
{71.03,137.53},{72.17,119.90},{19.90,54.96},{61.70,106.53},
{86.94,139.19},{93.06,150.70},{67.86,123.06},{67.03,122.25},
{62.82,110.61},{91.10,150.89},{98.67,143.84},{42.23,75.22},
{70.24,121.26},{65.04,134.36},{78.46,125.22},{22.79,59.19},
{68.07,119.57},{18.24,51.20},{30.87,81.60},{80.08,134.96},
{95.29,159.65},{39.22,82.84},{83.00,137.53},{23.58,55.76},
{78.09,151.92},{79.13,140.00},{42.00,89.66},{96.08,163.79},
{ 4.46,43.37},{18.44,57.17},{75.40,133.47},{24.25,68.80},
{34.40,77.65},{93.84,174.80},{55.93,108.84},{ 5.33,36.86},
{63.31,114.36},{ 6.81,25.12},{81.91,145.69},{24.79,74.42},
{ 4.50,33.52},{48.83,112.00},{72.52,129.48},{54.90,111.69},
{ 4.81,16.62},{28.60,76.66},{46.12,100.78},{55.86,105.53},
{22.15,48.61},{74.18,131.28},{92.63,149.47},{62.92,117.50},
{79.21,143.82},{46.16,93.44},{82.83,137.77},{70.08,110.10},
{16.87,44.10},{21.61,66.67},{69.91,114.17},{99.98,162.73},
{30.92,78.61},{29.27,76.68},{14.79,45.97},{73.25,117.73},
{98.62,162.89},{54.95,101.33},{98.80,161.82},{37.41,82.41},
{58.49,121.97},{73.98,143.53},{95.88,161.29},{33.85,59.12},
{92.79,152.49},{79.44,137.30},{92.52,141.39},{45.72,110.47},
{22.11,60.31},{99.53,161.01},{68.04,123.91},{16.78,52.36},
{15.63,44.17},{59.62,109.95},{58.89,113.04},{44.11,102.31},
{ 2.45,48.48},{87.73,159.89},{12.60,61.54},{64.39,126.19},
{67.78,122.29},{41.72,104.73},{16.16,43.45},{94.94,158.10},
{68.44,115.61},{47.97,87.70},{22.17,58.54},{ 9.27,32.20},
{18.84,60.44},{57.91,105.41},{ 6.04,37.37},{68.99,118.14},
{ 7.05, 4.68},{63.70,104.08},{58.15,102.31},{28.48,79.27},
{ 0.98,19.26},{54.54,106.97},{35.78,86.27},{40.13,97.09},
{14.06,41.38},{79.71,115.63},{77.10,138.44},{ 1.56,44.83},
{63.61,122.49},{40.04,87.80},{64.89,113.17},{96.64,154.10},
{96.30,159.43},{97.84,164.78},{74.04,136.76},{44.51,91.72},
{40.60,80.55},{83.38,141.40},{70.50,120.16},{76.64,119.86},
{43.64,99.97},{67.25,108.59},{61.51,112.69},{24.73,75.17},
{88.34,153.95},{85.83,140.98},{ 0.20,24.74},{78.95,139.67},
{75.31,151.22},{ 7.33,29.09},{24.41,52.45},{66.75,105.84},
{27.54,59.03},{30.84,79.05},{ 7.69,21.59},{94.77,157.28},
{53.51,93.25},{ 5.35,42.66},{ 4.27,27.74},{ 2.71,14.14},
{23.60,46.67},{43.63,59.86},{40.33,83.34},{60.91,103.18},
{14.53,49.11},{18.03,49.29},{33.38,75.47},{89.76,153.53},
{17.64,70.65},{17.11,43.02},{75.64,130.75},{12.80,40.92},
{12.47,31.14},{56.97,114.61},{32.38,75.70},{28.82,69.03},
{99.33,165.49},{81.46,147.33},{23.68,72.22},{73.55,125.49},
{22.55,60.90},{94.08,171.20},{17.58,37.35},{40.86,87.40},
{49.77,88.85},{83.15,139.80},{69.28,125.64},{17.49,47.84},
{33.16,87.43},{ 4.89,34.53},{80.15,137.39},{64.22,125.30},
{40.63,83.84},{80.15,151.77},{12.84,64.18},{29.96,66.67},
{26.36,52.66},{97.12,153.42},{42.58,68.36},{45.06,87.60},

{48.31,93.17},{33.63,73.95},{58.89,111.03},{47.37,70.24},
{60.02,103.43},{50.38,96.26},{18.19,54.87},{25.81,66.95},
{16.86,58.83},{38.56,88.49},{32.85,69.90},{89.85,163.12},
{65.50,111.34},{85.88,142.29},{ 7.95,38.29},{83.98,142.41},
{56.14,99.21},{99.19,162.94},{80.01,127.87},{86.31,134.95},
{59.34,99.92},{87.16,158.76},{ 3.91,27.31},{93.53,147.47},
{ 2.11,44.39},{69.16,134.50},{44.70,86.93},{70.54,130.95},
{41.87,85.29},{ 9.78,33.12},{50.76,82.35},{93.76,144.28},
{72.04,141.60},{38.98,86.87},{38.09,78.62},{96.90,146.78},
{70.20,116.03},{80.67,145.97},{10.27,45.64},{41.15,88.81},
{51.09,118.66},{67.66,120.42},{27.70,77.24},{ 3.02,15.19},
{32.03,80.76},{54.54,107.87},{86.35,144.18},{17.25,54.09},
{41.13,83.50},{56.44,108.74},{39.57,71.61},{ 2.12,21.67},
{29.36,73.62},{68.92,122.32},{76.30,120.73},{82.62,147.72},
{ 2.67,39.45},{33.70,73.72},{54.86,82.71},{98.28,149.14},
{26.97,57.08},{73.09,138.69},{31.72,71.68},{64.58,100.62},
{37.90,70.32},{89.64,150.03},{44.79,75.19},{20.26,55.72},
{ 0.40,18.79},{79.86,137.13},{34.03,77.00},{78.36,151.03},
{29.31,63.40},{46.21,87.28},{57.19,109.57},{17.20,55.41},
{78.13,123.06},{25.24,61.99},{13.73,47.05},{ 5.93,51.91},
{ 0.21,22.05},{83.43,126.58},{81.16,139.02},{19.17,61.02},
{61.99,99.25},{ 9.41,46.40},{41.57,93.92},{ 0.20,25.90},
{43.91,91.80},{60.37,108.89},{29.74,66.74},{20.39,67.92},
{80.19,141.75},{54.81,92.26},{26.95,58.42},{80.24,149.08},
{10.28,46.12},{54.56,91.13},{47.85,86.59},{83.75,153.73},
{76.30,136.05},{59.40,102.69},{77.22,139.53},{72.99,119.46},
{28.98,80.64},{83.22,132.99},{79.26,128.65},{75.64,117.61},
{55.05,117.36},{ 0.44,47.39},{60.93,95.98},{88.30,148.09},
{33.89,72.81},{69.12,113.19},{91.10,149.91},{64.26,125.49},
{87.91,151.18},{ 3.47,24.98},{68.19,122.08},{60.49,125.65},
{60.14,103.65},{94.01,151.82},{41.53,67.35},{43.66,69.84},
{25.44,62.62},{16.44,53.23},{50.12,102.14},{32.32,64.41},
{76.90,136.42},{ 7.94,30.08},{16.42,48.98},{66.64,114.34},
{28.52,70.75},{72.57,114.49},{ 2.10,43.57},{47.19,101.72},
{26.09,68.64},{87.07,164.83},{54.35,101.94},{64.33,115.05},
{26.57,60.29},{47.78,85.62},{81.82,149.78},{61.29,110.67},
{52.53,110.96},{81.40,133.00},{59.22,111.35},{58.51,107.63},
{66.82,105.32},{59.49,106.43},{59.09,100.05},{26.08,80.65},
{93.72,164.41},{34.92,80.88},{57.99,112.93},{23.84,71.22},
{ 3.66,25.27},{43.45,91.57},{31.94,82.81},{53.37,107.43},
{43.87,91.47},{57.85,115.10},{56.15,87.75},{79.27,119.97},
{91.75,143.96},{63.69,115.64},{78.77,143.60},{26.31,72.85},
{24.03,61.76},{16.25,40.68},{64.99,115.88},{53.07,92.62},
{45.53,80.91},{38.24,95.12},{87.68,150.73},{69.64,136.69},
{43.63,92.13},{53.59,97.48},{96.54,153.28},{78.42,147.16},
{47.61,81.21},{15.09,51.51},{ 5.47,27.31},{33.05,73.01},
{ 5.77,42.83},{82.27,140.11},{ 1.42,17.15},{67.19,132.58},
{73.48,131.34},{13.73,43.78},{45.37,84.68},{72.61,123.62},
{19.40,51.50},{48.05,81.15},{51.69,110.54},{10.96,35.75},
{ 4.02,33.25},{50.58,93.86},{54.31,100.10},{36.36,73.95},
{ 1.34,36.27},{33.61,74.56},{98.74,165.81},{91.12,149.43},

```
 {20.29,50.04},{ 1.43,16.08},{11.31,40.58},{80.95,150.57},
 { 5.44,30.05},{42.14,72.29},{15.45,63.26},{ 7.67,45.98},
 {65.03,114.83},{14.06,40.25},{56.68,90.67},{34.76,79.12},
 {67.72,133.60},{44.79,99.76},{28.77,75.44},{94.92,150.93},
 {94.05,151.68},{55.55,111.19},{25.70,54.71},{47.27,95.76},
 {53.63,79.83},{19.28,52.28},{52.99,90.72},{73.78,127.36},
 {73.66,137.23},{ 1.22,36.81},{86.39,145.93},{75.47,138.56},
 {74.18,130.21},{47.09,90.70},{74.38,133.83},{67.93,118.65},
 {27.75,75.81},{45.42,91.74},{49.36,114.95},{ 5.52,31.30},
 { 6.70,38.23},{94.46,161.36},{47.95,108.12},{70.55,118.63},
 {88.17,156.01},{ 8.42,41.84},{15.86,33.47},{37.95,66.31},
 {24.38,53.58},{65.78,110.63},{70.98,115.71},{96.42,186.30},
 {65.38,117.78},{34.11,85.81},{66.44,127.04},{50.25,77.33},
 {76.92,123.33},{79.41,142.18},{57.16,96.38},{98.39,169.75},
 {12.02,34.69},{91.79,132.83},{22.72,68.02},{33.75,75.89},
 {25.16,51.00},{53.47,91.68},{43.85,72.52},{65.47,126.99},
 {70.05,106.21},{17.54,61.73},{88.80,154.10},{67.63,134.52},
 { 7.76,22.07},{93.63,153.57},{40.29,87.39},{46.95,97.15},
 {27.39,79.84},{22.14,60.58},{90.50,150.75},{55.30,101.92},
 {94.50,173.63},{69.83,116.26},{76.92,140.02},{22.62,58.36}
};
```

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

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

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

```
sanjay@herald-OptiPlex-3050: ~/Downloads/linear_regression_1/linear_regression_3
File  Edit  View  Search  Terminal  Help
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression_1/linear_regression_3$
 cc -o linear_regression_multithread linear_regression_multithread.c -lm -pthrea
d
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression_1/linear_regression_3$
 cc -o linear_regression_multithread linear_regression_multithread.c -lm -pthrea
d
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression_1/linear_regression_3$
  chmod a+x mr.py
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression_1/linear_regression_3$
 ./mr.py ./linear_regression_multithread | grep Time
Time elapsed was 229150670ns or 0.229150670s
Time elapsed was 228175805ns or 0.228175805s
Time elapsed was 230551924ns or 0.230551924s
Time elapsed was 227279034ns or 0.227279034s
Time elapsed was 229163900ns or 0.229163900s
Time elapsed was 236041999ns or 0.236041999s
Time elapsed was 227863905ns or 0.227863905s
Time elapsed was 226566319ns or 0.226566319s
Time elapsed was 239533369ns or 0.239533369s
Time elapsed was 267099838ns or 0.267099838s
sanjay@herald-OptiPlex-3050:~/Downloads/linear_regression_1/linear_regression_3$
```

Write a short analysis of the results.

2 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

commit

efc710816ad0be76e8e7c04e5c011eb23fcbef19

Author: SanjayTamang <52417143+SanjayTamang@users.noreply.github.com>

Date:   Fri Nov 29 23:02:43 2019 +0545


   Add files via upload


commit

446b73e9f6c49cab44b12d84e090f1a2b41c1495

Author: SanjayTamang <52417143+SanjayTamang@users.noreply.github.com>

Date:   Fri Nov 29 22:08:25 2019 +0545


   Initial commit
```