# 6CS005 Learning Journal - Semester 1 2019/20

Put your name and student number here

## Table of Contents

# 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 Executing simple program of cracking 2 initial and 2 digit password for 10 times*

| | | | | |
|---|---|---|---|---|
| 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 |

Simple program of cracking 2 initial password was executed for 10 times to calculate mean running time that is 678.42 second.

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.

In this program we cracked 3 alphabet and 2 digit number. First we encrypted the password using SHA-512 algorithm. Proof of encrypting 3initial password is shown in figure below.

```
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ cc -o EncryptAAA99 EncryptAAA99.c -lcrypt
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ ./EncryptAAA99 ABC123
$6$KB$6S9dCIrK1jRJu0AoISAn1nWkUEnmDAGm/sKs1Vb1GJXhSrRJv0pe9jyRvM9ohg96sSQt9IPNPC6obgGTl6P2A1
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ ./EncryptAAA99 BER45
$6$KB$e1fG/f7Dr56YtgibdvLg7KpRb2bpCgGLv4eywNCmXXEgSV6jInDFCSkUoBq6Cn67b1GscDjxXExe9H3gudFgc.
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ ./EncryptAAA99 DAM89
$6$KB$mSVXI9v7J/9czxiCGGERm7BRZgkHaPo1TNZ8AqVdnkfz4INPofJQbRlGMpVzfFvFNIS5i6T5JLocC87tlG/NO/
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ LOL50
LOL50: command not found
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$ ./EncryptAAA99 LOL50
$6$KB$FsCnEgeVl5KRBe62eF4R3mxZORQHEVeBzPXartqdZsZ3rfvLHgfQhEZtQM5xEFuqrPLN5h2/0DCy1F41oRFGv0
asmit@asmit-Aspire:~/Desktop/POSIX/Password_Cracking/Three_Initial$
```

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 678*26 =17,628 second time for completing 3 initial program.

Calculation,

Estimated time= original time * 26 = 678 *26=17,628 second

```
1. #include <stdio.h>
2. #include <string.h>
3. #include <stdlib.h>
4. #include <crypt.h>
5. #include <time.h>
6.
7.
8.
9.
10.  int n_passwords = 4;
11.
```

```
12.  char *encrypted_passwords[] = {
13.    "$6$KB$6S9dCIrK1jRJu0AoISAn1nWkUEnmDAGm/sKs1Vb1GJXhSrRJv0pe9jyRvM9ohg96sSQt9IPNPC6obgGTl6P2A1",
14.    "$6$KB$e1fG/f7Dr56YtgibdvLg7KpRb2bpCgGLv4eywNCmXXEgSV6jInDFCSkUoBq6Cn67b1GscDjxXExe9H3gudFgc.",
15.    "$6$KB$mSVXI9v7J/9czxiCGGERm7BRZgkHaPo1TNZ8AqVdnkfz4INPofJQbRlGMpVzfFvFNIS5i6T5JLocC87tlG/NO/",
16.    "$6$KB$FsCnEgeVl5KRBe62eF4R3mxZORQHEVeBzPXartqdZsZ3rfvLHgfQhEZtQM5xEFuqrPLN5h2/0DCy1F41oRFGv0"
17.  };
18.
19.  /**
20.   Required by lack of standard function in C.
21.  */
22.
23.  void substr(char *dest, char *src, int start, int length){
24.    memcpy(dest, src + start, length);
25.    *(dest + length) = '\0';
26.  }
27.
28.  /**
29.   This function can crack the kind of password explained above. All combinations
30.   that are tried are displayed and when the password is found, #, is put at the
31.   start of the line. Note that one of the most time consuming operations that
32.   it performs is the output of intermediate results, so performance experiments
33.   for this kind of program should not include this. i.e. comment out the printfs.
34.  */
35.
36.  void Passwordcrack(char *salt_and_encrypted){
37.    int x, y, z, a;     // Loop counters
38.    char salt[7];    // String used in hashing the password. Need space for \0
39.    char plain[7];   // The combination of letters currently being checked
40.    char *enc;       // Pointer to the encrypted password
41.    int count = 0;   // The number of combinations explored so far
42.
43.    substr(salt, salt_and_encrypted, 0, 6);
44.
45.    for(x='A'; x<='Z'; x++){
46.      for(y='A'; y<='Z'; y++){
47.          for(z='A'; z<='Z'; z++){
48.          for(a=0; a<=99; a++){
49.                  sprintf(plain, "%c%c%c%02d", x, y, z,a);
50.                  enc = (char *) crypt(plain, salt);
51.                  count++;
52.                  if(strcmp(salt_and_encrypted, enc) == 0){
53.                    printf("#%-8d%s %s\n", count, plain, enc);
54.                  } /*else {
```

```
55.                   printf(" %-8d%s %s\n", count, plain, enc);
56.                 }*/
57.         }
58.           }
59.       }
60.    }
61.    printf("%d solutions explored\n", count);
62.  }
63.
64.  //Calculating time
65.
66.  int time_difference(struct timespec *start, struct timespec *finish, long long int *difference)
67.   {
68.            long long int ds =  finish->tv_sec - start->tv_sec;
69.            long long int dn =  finish->tv_nsec - start->tv_nsec;
70.
71.            if(dn < 0 ) {
72.              ds--;
73.              dn += 1000000000;
74.    }
75.            *difference = ds * 1000000000 + dn;
76.            return !(*difference > 0);
77.  }
78.  int main(int argc, char *argv[])
79.  {
80.          int i;
81.          struct timespec start, finish;
82.          long long int time_elapsed;
83.
84.          clock_gettime(CLOCK_MONOTONIC, &start);
85.
86.          for(i=0;i<n_passwords;i<i++)
87.          {
88.                  Passwordcrack(encrypted_passwords[i]);
89.          }
90.          clock_gettime(CLOCK_MONOTONIC, &finish);
91.            time_difference(&start, &finish, &time_elapsed);
92.            printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
93.                                                  (time_elapsed/1.0e9));
94.    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 2 Three initial password cracking*

In my hypothesis I said that it will take about 17,628 second to complete 3 initial program but it took about 16794.49 second.

Estimated time=17,628 second. (4.89 hrs)
Time difference = 17,628 -16,794.49 =834 second (13.9 min)

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

We executed simple program and multithread program for 10 times and calculated mean running 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 |

Mean running time of simple program is 678425696637.6 second.

| | | |
|---|---|---|
| Time elapsed was | 88 | ns |
| Time elapsed was | 102 | ns |
| Time elapsed was | 111 | ns |
| Time elapsed was | 134 | ns |
| Time elapsed was | 107 | ns |
| Time elapsed was | 64 | ns |
| Time elapsed was | 88 | ns |
| Time elapsed was | 65 | ns |
| Time elapsed was | 90 | ns |
| Time elapsed was | 74 | ns |
| total | 923 | ns |
| mean | 92.3 | ns |
| Average mean | 46.15 | ns |
| Time in minute | 0.769167 | ns |

Mean running of multithreading program is 92.3 nano second.

In this program we created two threads (kernel_function_1 and kernel_function_2) to work on the program where the first thread cracks the password from the letter A to M and the second thread cracks the password from letter N to Z. Both thread will run Simultaneous explore the possible password. Which results to the mean time of this program was used was reduced to half which was proved by comparing above tables.

```
1. #include <stdio.h>
2. #include <string.h>
3. #include <stdlib.h>
4. #include <crypt.h>
5. #include <time.h>
6. #include <pthread.h>
7.
8. int n_passwords = 4;
9.
10.  char *encrypted_passwords[] = {
11.  "$6$KB$0G24VuNaA9ApVG4z8LkI/OOr9a54nBfzgQjbebhqBZxMHNg0HiYYf1Lx/HcGg6q1nnOSArPtZYbGy7yc5V.wP/",
12.  "$6$KB$VDUCASt5S88l82JzexhKDQLeUJ5zfxr16VhlVwNOs0YLiLYDciLDmN3QYAE80UIzfryYmpR.NFmbZvAGNoaHW.",
13.  "$6$KB$0n1YjoLnJBuAdeBsYFW3fpZzMPP8xycQbEj35GvoerMnEkWIAKnbUBAb70awv5tfHylWkVzcwzHUNy/7l7I1c/",
14.  "$6$KB$HKffNNiGzngqYueF89z3gwWZMg.xUBIz/00QSCbgwKtRHmwUbZX6jTH4VUAg3L3skaO8qtNf5LE7WP39jQ7ZJ0"
15.  };
16.
17.
18.  void substr(char *dest, char *src, int start, int length){
19.    memcpy(dest, src + start, length);
20.    *(dest + length) = '\0';
21.  }
22.
23.  /**
24.   This function can crack the kind of password explained above. All
25.  combinations
26.   that are tried are displayed and when the password is found, #, is put
27.  at the
28.   start of the line. Note that one of the most time consuming operations
29.  that
30.   it performs is the output of intermediate results, so performance
31.  experiments
32.   for this kind of program should not include this. i.e. comment out the
33.  printfs.
34.  */
35.
```

```
36.  void posix()
37.  {
38.    int i;
39.  pthread_t thread1, thread2;
40.
41.      void *kernel_function_1();
42.      void *kernel_function_2();
43.  for(i=0;i<n_passwords;i<i++) {
44.
45.
46.      pthread_create(&thread1, NULL,kernel_function_1, encrypted_passwords[i]);
47.      pthread_create(&thread2, NULL,kernel_function_2, encrypted_passwords[i]);
48.
49.      pthread_join(thread1, NULL);
50.      pthread_join(thread2, NULL);
51.  pthread_exit(&thread1);
52.  pthread_exit(&thread2);
53.
54.   }
55.  }
56.
57.  void *kernel_function_1(char *salt_and_encrypted){
58.    int x, y, z;     // Loop counters
59.    char salt[7];
60.    char plain[7];   // The combination of letters currently being checked
61.    char *enc;       // Pointer to the encrypted password
62.    int count = 0;   // The number of combinations explored so far
63.
64.  substr(salt, salt_and_encrypted, 0, 6);
65.
66.  for(x='A'; x<='M'; x++){
67.     for(y='A'; y<='Z'; y++){
68.       for(z=0; z<=99; z++){
69.          sprintf(plain, "%c%c%02d", x, y, z);
70.          enc = (char *) crypt(plain, salt);
71.          count++;
72.          if(strcmp(salt_and_encrypted, enc) == 0){
73.            printf("#%-8d%s %s\n", count, plain, enc);
74.          }
75.        }
76.      }
77.    }
78.    printf("%d solutions explored\n", count);
```

```
79.  }
80.
81.  void *kernel_function_2(char *salt_and_encrypted){
82.    int i, j, k;      // Loop counters
83.    char salt[7];     // String used in hahttps://www.youtube.com/watch?v=L8yJjIGleMwshing the password. Need space
84.    char plain[7];    // The combination of letters currently being checked
85.    char *enc;        // Pointer to the encrypted password
86.    int count = 0;    // The number of combinations explored so far
87.
88.    substr(salt, salt_and_encrypted, 0, 6);
89.
90.    for(i='N'; i<='Z'; i++){
91.      for(j='A'; j<='Z'; j++){
92.        for(k=0; k<=99; k++){
93.          sprintf(plain, "%c%c%02d", i,j,k);
94.          enc = (char *) crypt(plain, salt);
95.          count++;
96.          if(strcmp(salt_and_encrypted, enc) == 0){
97.            printf("#%-8d%s %s\n", count, plain, enc);
98.          }
99.        }
100.     }
101.   }
102.   printf("%d solutions explored\n", count);
103. }
104.
105. //Calculating time
106.
107. int time_difference(struct timespec *start, struct timespec *finish, long long int *difference)
108.  {
109.           long long int ds =  finish->tv_sec - start->tv_sec;
110.           long long int dn =  finish->tv_nsec - start->tv_nsec;
111.
112.           if(dn < 0 ) {
113.             ds--;
114.             dn += 1000000000;
115.   }
116.           *difference = ds * 1000000000 + dn;
117.           return !(*difference > 0);
118. }
119. int main(int argc, char *argv[])
120. {
121.
```

```
122.         struct timespec start, finish;
123.         long long int time_elapsed;
124.         posix();
125.         clock_gettime(CLOCK_MONOTONIC, &start);
126.         clock_gettime(CLOCK_MONOTONIC, &finish);
127.           time_difference(&start, &finish, &time_elapsed);
128.           printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
129.                                                 (time_elapsed/1.0e9));
130.   return 0;
131. }
```

## 1.2 Image Processing

Insert the image displayed by your program


```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <GL/glut.h>
5. #include <GL/gl.h>
6. #include <malloc.h>
7. #include <signal.h>
8. #include <pthread.h>
9.
10.  #define width 100
11.  #define height 72
12.  typedef struct arg_t{
13.  int start;
14.  int stride;
```

```
15.  }arg_t;
16.
17.  unsigned char image[], results[width * height];
18.
19.  void edges(unsigned char *in, unsigned char *out,arg_t *args) {
20.    int i;
21.    int n_pixels = width * height;
22.
23.    for(i=args->start;i<n_pixels;i+=args->stride) {
24.      int x, y; // the pixel of interest
25.      int b, d, f, h; // the pixels adjacent to x,y used for the calculation
26.      int r; // the result of calculate
27.
28.      y = i / width;
29.      x = i - (width * y);
30.
31.      if (x == 0 || y == 0 || x == width - 1 || y == height - 1) {
32.        results[i] = 0;
33.      } else {
34.        b = i + width;
35.        d = i - 1;
36.        f = i + 1;
37.        h = i - width;
38.
39.        r = (in[i] * 4) + (in[b] * -1) + (in[d] * -1) + (in[f] * -1)
40.            + (in[h] * -1);
41.
42.        if (r > 0) { // if the result is positive this is an edge pixel
43.          out[i] = 255;
44.        } else {
45.          out[i] = 0;
46.        }
47.      }
48.    }
49.  }
50.
51.  void *Detection(void *args)
52.  {
53.  edges(image,results,args);
54.  }
55.
56.
57.  void tidy_and_exit() {
```

```
58.    exit(0);
59.  }
60.
61.  void sigint_callback(int signal_number){
62.    printf("\nInterrupt from keyboard\n");
63.    tidy_and_exit();
64.  }
65.
66.  static void show() {
67.    glClear(GL_COLOR_BUFFER_BIT);
68.    glRasterPos4i(-1, -1, 0, 1);
69.    glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, image);
70.    glRasterPos4i(0, -1, 0, 1);
71.    glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, results);
72.    glFlush();
73.  }
74.
75.  static void key_pressed(unsigned char key, int x, int y) {
76.    switch(key){
77.      case 27: // escape
78.        tidy_and_exit();
79.        break;
80.      default:
81.        printf("\nPress escape to exit\n");
82.        break;
83.    }
84.  }
85.  int time_difference(struct timespec *start, struct timespec *finish,
86.                      long long int *difference) {
87.    long long int ds =  finish->tv_sec - start->tv_sec;
88.    long long int dn =  finish->tv_nsec - start->tv_nsec;
89.
90.    if(dn < 0 ) {
91.      ds--;
92.      dn += 1000000000;
93.    }
94.    *difference = ds * 1000000000 + dn;
95.    return !(*difference > 0);
96.  }
97.  int main(int argc, char **argv) {
98.    signal(SIGINT, sigint_callback);
99.   glutInit(&argc, argv);
100. struct timespec start, finish;
```

```
101.   long long int time_elapsed;
102.
103.   clock_gettime(CLOCK_MONOTONIC, &start);
104.
105.
106.
107.   clock_gettime(CLOCK_MONOTONIC, &finish);
108.   time_difference(&start, &finish, &time_elapsed);
109.   printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
110.          (time_elapsed/1.0e9));
111.
112.   printf("image dimensions %dx%d\n", width, height);
113.  pthread_t t1, t2, t3, t4;
114.
115.   arg_t t1_arguments;
116.   t1_arguments.start = 0;
117.   t1_arguments.stride = 4;
118.
119.   arg_t t2_arguments;
120.   t2_arguments.start = 1;
121.   t2_arguments.stride = 4;
122.
123.   arg_t t3_arguments;
124.   t3_arguments.start = 2;
125.   t3_arguments.stride = 4;
126.
127.   arg_t t4_arguments;
128.   t4_arguments.start = 3;
129.   t4_arguments.stride = 4;
130.
131.   void *Detection();
132.
133.   pthread_create(&t1, NULL, Detection, &t1_arguments);
134.   pthread_create(&t2, NULL, Detection, &t2_arguments);
135.   pthread_create(&t3, NULL, Detection, &t3_arguments);
136.   pthread_create(&t4, NULL, Detection, &t4_arguments);
137.
138.   pthread_join(t1, NULL);
139.   pthread_join(t2, NULL);
140.   pthread_join(t3, NULL);
141.   pthread_join(t4, NULL);
142.
143.
```

```
144.
145.   glutInitWindowSize(width * 2,height);
146.   glutInitDisplayMode(GLUT_SINGLE | GLUT_LUMINANCE);
147.
148.   glutCreateWindow("6CS005 Image Progessing Courework");
149.   glutDisplayFunc(show);
150.   glutKeyboardFunc(key_pressed);
151.   glClearColor(0.0, 1.0, 0.0, 1.0);
152.
153.   glutMainLoop();
154.
155.   tidy_and_exit();
156.  pthread_exit(&t1);
157.   pthread_exit(&t2);
158.   pthread_exit(&t3);
159.   pthread_exit(&t4);
160.
161.   return 0;
162. }
163.
164. unsigned char image[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
165.   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
166.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
167.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
168.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
169.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
170.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
171.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
172.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
173.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
174.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
175.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
176.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
177.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
178.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
179.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
180.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
181.   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
182.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
183.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
184.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
185.   0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
186.   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
```

```
187.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
188.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
189.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
190.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
191.   0,0,0,0,0,0,0,255,255,0,0,0,0,255,255,255,255,255,255,
192.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
193.   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
194.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
195.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
196.   0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,255,255,
197.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
198.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
199.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
200.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,0,
201.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
202.   255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
203.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
204.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
205.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
206.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
207.   0,255,255,0,255,255,255,0,0,255,255,255,255,255,255,255,255,255,255,255,
208.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
209.   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
210.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
211.   0,0,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
212.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,
213.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
214.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
215.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
216.   0,0,0,0,0,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
217.   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,0,
218.   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
219.   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
220.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
221.   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,0,
222.   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
223.   255,255,255,255,255,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
224.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
225.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
226.   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
227.   255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
228.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,255,255,
229.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
```

```
230.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
231.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
232.   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,255,
233.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
234.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
235.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
236.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
237.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
238.   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
239.   255,255,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
240.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
241.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
242.   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
243.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
244.   255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,255,255,255,255,
245.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
246.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
247.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
248.   255,255,255,255,255,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
249.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,255,255,
250.   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
251.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
252.   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
253.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,255,255,255,255,255,255,
254.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255.   255,255,0,0,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
256.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
257.   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
258.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,
259.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
260.   255,255,255,255,255,255,255,255,0,0,255,255,0,255,255,255,255,0,0,0,
261.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
262.   0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,255,255,255,
263.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
264.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
265.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,
266.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
267.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
268.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
269.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
270.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
271.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
272.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
273.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
274.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
275.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
276.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
277.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
278.   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
279.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
280.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
281.   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
282.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
283.   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
284.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
285.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
286.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
287.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
288.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
289.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
290.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
291.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
292.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
293.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
294.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
295.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
296.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
297.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
298.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
299.   0,0,255,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
300.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
301.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
302.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
303.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
304.   0,0,0,0,0,255,255,0,0,0,255,255,255,255,255,255,255,255,255,255,
305.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
306.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
307.   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
308.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
309.   0,0,0,0,0,0,0,0,0,255,0,0,0,0,255,255,255,255,255,
310.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
311.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
312.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
313.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
314.   0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,
315.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
```

```
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,255,0,0,0,0,0,0,0,255,255,255,255,255,
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,
0,0,0,0,0,0,0,255,0,0,255,255,255,0,0,0,0,0,0,
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,
255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
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,
255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,0,0,0,0,0,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,
0,0,255,255,255,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,255,255,255,255,0,255,0,255,255,255,0,0,0,0,0,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,0,0,0,
0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,255,0,0,255,255,255,255,255,255,255,255,0,0,255,255,255,0,0,
0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,255,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
```

```
359.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
360.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
361.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
362.   0,255,255,0,0,255,0,255,255,255,255,255,255,255,255,255,255,0,0,
363.   0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
364.   0,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
365.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
366.   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
367.   255,255,255,255,255,0,255,255,255,0,0,0,0,0,255,255,255,255,255,
368.   0,0,255,255,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
369.   0,0,0,0,0,0,255,255,255,255,255,255,255,0,0,0,0,0,0,
370.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
371.   0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
372.   255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,0,0,0,0,0,
373.   255,255,0,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,
374.   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,0,255,0,0,
375.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
376.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,255,
377.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,
378.   255,0,0,0,0,255,255,0,0,0,0,0,255,0,0,0,0,0,0,
379.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,
380.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
381.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
382.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
383.   255,255,255,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
384.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
385.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
386.   0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
387.   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
388.   255,255,255,255,255,255,255,255,0,255,255,255,0,0,0,0,0,0,255,
389.   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
390.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
391.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
392.   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
393.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,
394.   0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
395.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
396.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
397.   0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
398.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
399.   0,0,0,255,255,0,0,255,0,0,0,0,0,0,0,0,0,0,0,
400.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
401.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
402.   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
403.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
404.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
405.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
406.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
407.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
408.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
409.   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
410.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
411.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
412.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
413.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
414.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
415.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
416.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
417.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
418.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
419.   255,255,0,255,255,255,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
420.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
421.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
422.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
423.   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
424.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,255,255,255,
425.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
426.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
427.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
428.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
429.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
430.   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
431.   0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,
432.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
433.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
434.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
435.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
436.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
437.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
438.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
439.   255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
440.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
441.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
442.   0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
443.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
444.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
```

```
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,
0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
0,0,0,255,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,
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,
```

```
488.   0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
489.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
490.   0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
491.   255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,255,
492.   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
493.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
494.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
495.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
496.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
497.   0,0,0,0,0,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
498.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
499.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
500.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
501.   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
502.   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,0,0,
503.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
504.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
505.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
506.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
507.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,
508.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
509.   0,0,0,0,0,0,255,0,0,0,0,0,0,0,255,0,0,0,0,
510.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
511.   0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
512.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
513.   0,0,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
514.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
515.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
516.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
517.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
518.   0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
519.   0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
520.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
521.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
522.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
523.   255,255,255,255,255,0,0,0,0,0,255,0,0,0,0,0,0,255,0,
524.   0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,
525.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
526.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
527.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
528.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
529.   0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,
530.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
531.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
532.   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
533.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,
534.   0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,0,0,0,0,
535.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
536.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
537.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
538.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
539.   255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,
540.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
541.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
542.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
543. };
```

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$
```

```
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$
```

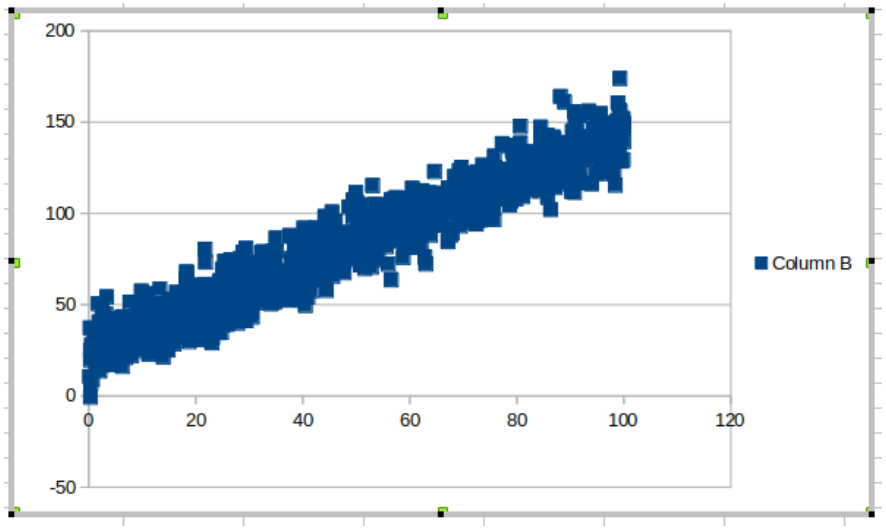
| Simple programe | | | | | Multithread program | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Time elaspsed was | 83 | ns or | 0.000000083 | s | Time elapsed was | 63 | ns | 0.000000063 | s |
| Time elaspsed was | 129 | ns or | 0.000000129 | s | Time elapsed was | 48 | ns | 0.000000048 | s |
| 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 |

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

Above table shows the mean running time of original and multithreading program of image processing. Comparing the table we recognize that the program which used multithreading program take less time that of simple program. This is because the multithreading program improves the performance and concurrency. Mean running time of simple program is 9.3 second and for multithread program is 6.12 second.

## 1.3 Linear Regression

Insert a scatter plot of your data.



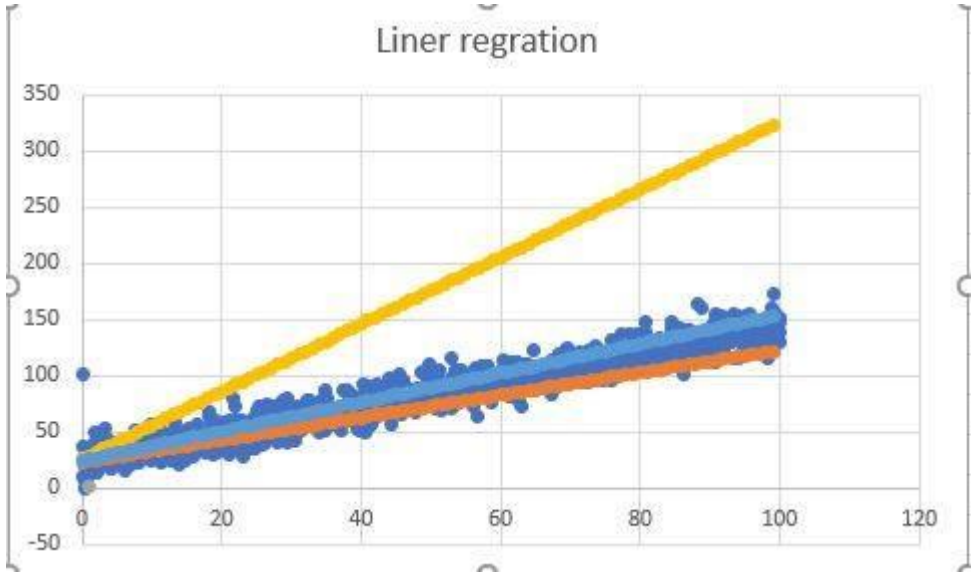
Have 3 guesses at the optimum values for m and c and present them in a graph that overlays your data.

Beside optimum value for m and c we have guessed the 3 other value for m and c which are shown in the figure below.

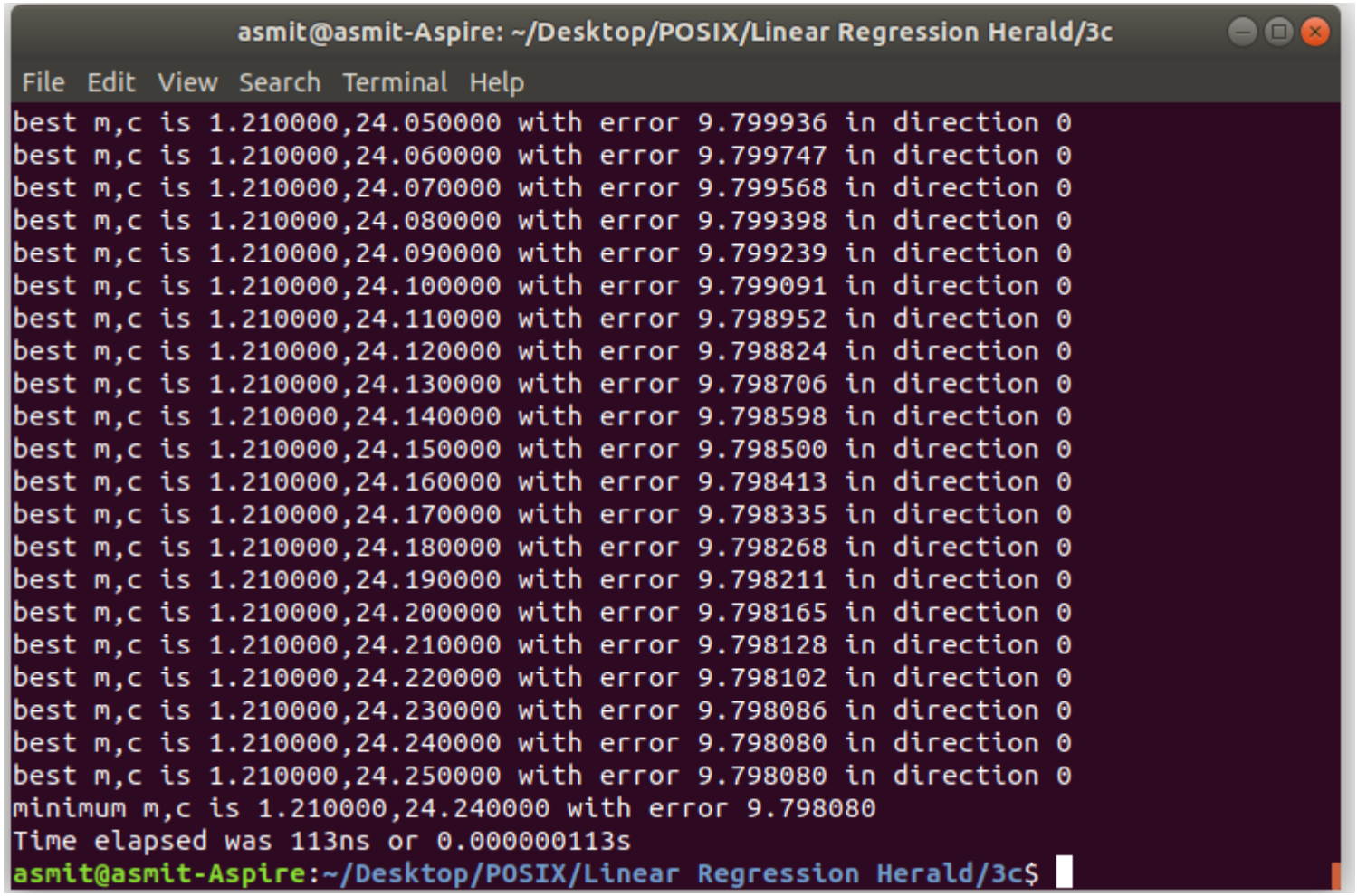
```
asmit@asmit-Aspire:~/Desktop/POSIX/Linear Regression Herald/3b$ ./linearReg01 1.20 24.56
0.00,24.56
1.00,25.76
2.00,26.96
3.00,28.16
```

```
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
```

```
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
```


Liner regration
350
300
250
200
150
100
50
0
-50
0
20
40
60
80
100
120

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



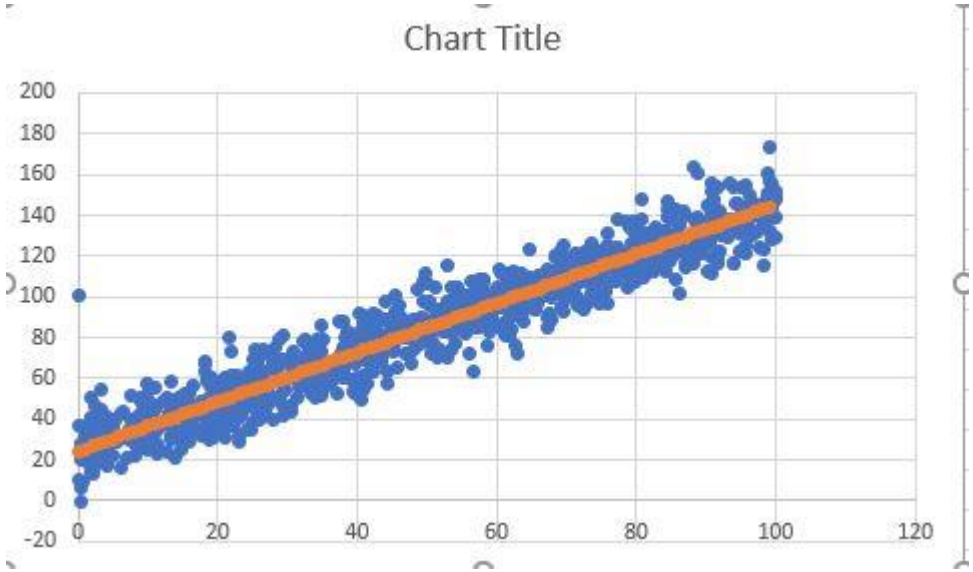



*Figure 3 solution*

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

```
1. #include <stdio.h>
2. #include <math.h>
3. #include <time.h>
4. #include <pthread.h>
5. #include <string.h>
6. #include <stdlib.h>
7.
8.
9. int i;
10.    double bm = 1.3;
11.    double bc = 10;
12.    double be;
13.    double dm[8];
14.    double dc[8];
15.    double e[8];
16.    double step = 0.01;
17.    double best_error = 999999999;
18.    int best_error_i;
19.    int minimum_found = 0;
20.
21.    double om[] = {0,1,1, 1, 0,-1,-1,-1};
22.    double oc[] = {1,1,0,-1,-1,-1, 0, 1};
23.
24.  typedef struct point_t {
25.    double x;
26.    double y;
27.  } point_t;
28.
29.  int n_data = 1000;
30.  point_t data[];
31.
32.  double residual_error(double x, double y, double m, double c) {
33.    double e = (m * x) + c - y;
34.    return e * e;
35.  }
36.
37.  double rms_error(double m, double c) {
38.    int i;
39.    double mean;
40.    double error_sum = 0;
41.
```

```
42.    for(i=0; i<n_data; i++) {
43.      error_sum += residual_error(data[i].x, data[i].y, m, c);
44.    }
45.
46.    mean = error_sum / n_data;
47.
48.    return sqrt(mean);
49.  }
50.
51.  int time_difference(struct timespec *start, struct timespec *finish, long long int *difference)
52.   {
53.            long long int ds =  finish->tv_sec - start->tv_sec;
54.            long long int dn =  finish->tv_nsec - start->tv_nsec;
55.
56.            if(dn < 0 ) {
57.              ds--;
58.              dn += 1000000000;
59.    }
60.            *difference = ds * 1000000000 + dn;
61.            return !(*difference > 0);
62.  }
63.
64.  void *linear_regression(void *args){
65.          int *a=args;
66.          int i=*a;
67.
68.          printf("\n i in thread fun=%d",i);
69.                  dm[i] = bm + (om[i] * step);
70.                  dc[i] = bc + (oc[i] * step);
71.
72.                  e[i] = rms_error(dm[i], dc[i]);
73.                  if(e[i] < best_error) {
74.                  best_error = e[i];
75.                  best_error_i = i;
76.                  pthread_exit(NULL);
77.
78.                  }
79.          }
80.
81.  int main() {
82.          struct timespec start, finish;
83.          long long int time_elapsed;
84.
```

```
85.          clock_gettime(CLOCK_MONOTONIC, &start);
86.
87.          int i;
88.          pthread_t p_threads[8];
89.
90.          be = rms_error(bm, bc);
91.
92.          while(!minimum_found) {
93.          for(i=0;i<8;i++) {
94.                  pthread_create(&p_threads[i],NULL,linear_regression,&i);
95.                  pthread_join(p_threads[i],NULL);
96.          }
97.          printf("best m,c is %lf,%lf with error %lf in direction %d\n",
98.          dm[best_error_i], dc[best_error_i], best_error, best_error_i);
99.          if(best_error < be) {
100.                  be = best_error;
101.                 bm = dm[best_error_i];
102.                 bc = dc[best_error_i];
103.          } else {
104.       minimum_found = 1;
105.                 }
106.         }
107.       printf("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);
108.
109.         clock_gettime(CLOCK_MONOTONIC, &finish);
110.           time_difference(&start, &finish, &time_elapsed);
111.           printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
112.                                                 (time_elapsed/1.0e9));
113.         pthread_exit(NULL);
114.          return 0;
115.
116. }
117.
118. point_t data[] = {
119.   {72.12,100.78},{65.40,107.86},{82.27,131.60},{82.31,122.34},
120.   {89.41,121.50},{71.37,113.51},{82.62,112.38},{69.57,102.96},
121.   {65.38,99.27},{84.50,138.85},{87.18,114.17},{73.03,109.21},
122.   {67.26,102.06},{72.25,113.23},{61.28,101.59},{41.60,84.24},
123.   {40.14,57.03},{15.24,45.58},{61.88,89.90},{34.89,72.77},
124.   { 8.91,36.34},{30.45,46.18},{67.93,89.35},{68.82,112.80},
125.   {63.96,99.32},{32.36,56.12},{42.20,63.66},{24.47,60.75},
126.   { 1.96,28.62},{41.42,68.41},{34.49,73.14},{ 8.03,22.13},
127.   {80.55,117.79},{85.54,130.80},{68.99,103.13},{99.32,144.79},
```

```
128.   {91.71,153.61},{71.17,108.40},{85.28,120.11},{99.52,128.68},
129.   {13.24,31.67},{ 5.19,40.15},{ 9.84,57.36},{29.42,54.01},
130.   {89.68,126.25},{29.45,41.30},{79.63,132.59},{71.88,107.31},
131.   {20.05,48.38},{40.98,54.11},{56.55,63.61},{77.22,114.17},
132.   {63.86,88.10},{92.93,134.84},{56.84,101.20},{34.31,71.18},
133.   {93.89,116.43},{38.02,63.78},{61.25,94.71},{71.02,103.42},
134.   {95.05,142.82},{96.24,133.50},{19.50,50.92},{41.14,70.59},
135.   {91.49,134.05},{54.05,98.31},{36.59,68.48},{91.14,130.45},
136.   {44.76,88.98},{77.28,138.16},{64.80,96.33},{43.25,70.08},
137.   {55.55,95.70},{ 3.77,39.03},{ 3.23,44.69},{86.72,127.42},
138.   {84.62,131.54},{26.13,71.24},{61.22,98.22},{53.90,96.07},
139.   {64.81,109.35},{91.66,116.79},{53.65,104.81},{38.42,66.16},
140.   {62.33,112.41},{ 7.41,29.86},{41.59,57.59},{56.49,91.60},
141.   {15.94,42.82},{97.46,140.29},{57.17,85.11},{26.94,45.86},
142.   {73.14,96.37},{18.61,60.58},{15.69,44.16},{20.79,33.86},
143.   {65.02,106.03},{38.09,72.71},{87.15,116.68},{77.45,123.08},
144.   {90.47,126.33},{26.80,44.96},{75.94,119.76},{33.83,69.11},
145.   {63.59,103.98},{38.05,72.36},{68.28,110.76},{ 3.34,54.22},
146.   {45.40,92.84},{78.37,113.49},{27.11,46.46},{32.32,68.44},
147.   {20.97,30.90},{37.92,75.11},{96.85,130.96},{69.40,95.17},
148.   { 3.29,30.06},{64.41,103.44},{15.80,52.64},{61.76,97.79},
149.   { 1.62,33.98},{29.03,58.02},{18.74,34.93},{25.41,73.73},
150.   {28.78,65.94},{14.64,50.31},{82.85,133.70},{41.62,90.32},
151.   {99.28,144.95},{90.16,133.18},{40.45,77.72},{ 1.79,50.44},
152.   {31.80,62.71},{26.30,40.89},{47.57,83.15},{17.78,44.90},
153.   {69.48,93.13},{87.98,126.95},{69.84,106.00},{37.06,61.61},
154.   {90.65,133.97},{10.73,46.60},{38.84,79.90},{ 4.75,33.89},
155.   {48.99,89.31},{ 2.51,47.09},{34.99,86.40},{29.79,54.52},
156.   {91.30,133.72},{74.12,122.86},{90.93,141.88},{51.14,89.93},
157.   {84.53,142.49},{26.84,58.79},{ 6.95,20.98},{49.80,85.14},
158.   {22.82,57.02},{44.08,89.32},{22.28,48.72},{21.12,50.68},
159.   {65.69,93.93},{27.84,39.97},{ 1.92,40.39},{ 9.36,33.54},
160.   {88.10,123.02},{18.15,63.84},{21.80,39.76},{64.42,101.03},
161.   { 2.23,22.52},{55.68,99.56},{37.55,87.77},{74.23,104.87},
162.   {11.96,37.30},{23.60,45.84},{11.13,34.32},{ 9.05,48.79},
163.   {56.11,100.21},{19.31,54.44},{ 6.27,16.17},{64.65,101.39},
164.   {50.25,77.59},{69.33,95.12},{47.52,87.79},{28.97,65.98},
165.   {71.56,95.30},{19.71,41.47},{57.66,96.65},{41.07,74.10},
166.   {35.08,79.46},{40.80,87.01},{ 0.31,19.82},{90.78,111.55},
167.   {34.39,72.03},{99.97,139.40},{30.86,73.03},{14.37,50.15},
168.   { 6.11,42.76},{21.75,80.30},{89.94,127.56},{10.86,42.40},
169.   {13.07,42.98},{84.47,147.14},{83.44,132.18},{32.24,63.57},
170.   {66.93,102.41},{34.48,68.96},{ 3.46,22.82},{94.84,130.83},
```

```
171.   {49.41,107.26},{71.64,99.82},{47.28,80.62},{39.17,68.77},
172.   {58.05,108.35},{69.27,109.81},{47.64,73.34},{34.64,73.15},
173.   {22.86,46.34},{37.76,66.19},{ 3.12,39.11},{60.59,111.05},
174.   {91.99,122.76},{96.60,138.86},{ 3.58,23.35},{22.81,60.18},
175.   {13.93,21.32},{69.51,106.41},{19.57,43.39},{79.11,115.68},
176.   {80.89,124.36},{44.42,57.78},{33.28,73.04},{21.45,49.88},
177.   {70.57,113.77},{45.63,65.60},{55.99,72.21},{21.62,41.47},
178.   {61.74,98.99},{ 9.30,29.77},{75.32,106.74},{27.97,73.44},
179.   {74.77,115.98},{42.93,82.67},{92.32,138.05},{25.55,64.34},
180.   { 0.48,23.51},{79.52,111.52},{52.83,70.58},{51.45,87.28},
181.   {62.72,90.41},{ 4.16,40.60},{70.13,115.25},{55.96,97.34},
182.   {93.88,154.09},{46.21,90.04},{34.75,51.46},{54.45,89.56},
183.   {80.69,129.36},{45.14,73.00},{47.34,85.69},{70.16,118.02},
184.   { 4.26,17.14},{61.56,98.04},{15.95,28.56},{74.06,118.48},
185.   {65.29,99.71},{19.08,55.64},{37.82,72.36},{58.22,103.93},
186.   {50.52,82.15},{26.25,60.91},{97.77,123.91},{39.13,68.03},
187.   {15.09,41.88},{32.61,61.64},{11.23,22.85},{61.92,98.02},
188.   {73.63,126.32},{35.12,54.74},{12.98,42.69},{83.87,128.60},
189.   {45.65,78.81},{42.85,90.57},{76.74,117.53},{19.05,49.60},
190.   {69.03,104.16},{23.66,54.97},{52.85,85.94},{82.07,128.27},
191.   {74.77,111.22},{95.04,136.69},{40.49,49.53},{ 4.16,28.40},
192.   { 7.69,51.29},{29.37,80.82},{86.06,122.19},{ 3.92,23.24},
193.   {62.76,108.89},{27.12,54.24},{10.24,33.84},{79.86,107.97},
194.   {57.09,85.27},{10.29,54.38},{53.50,82.98},{12.83,50.29},
195.   { 2.09,13.69},{88.73,135.16},{42.72,87.10},{40.20,91.88},
196.   {40.10,76.49},{80.22,133.65},{57.55,93.99},{29.34,69.08},
197.   { 2.90,41.26},{44.60,82.03},{47.93,89.05},{98.17,123.11},
198.   {17.21,45.91},{42.37,79.83},{90.89,119.42},{ 7.81,36.64},
199.   {76.14,123.86},{47.79,83.40},{95.27,144.30},{44.13,98.20},
200.   {19.97,37.36},{90.66,131.96},{75.41,117.80},{57.14,107.91},
201.   {25.92,41.69},{90.86,130.36},{44.78,79.02},{23.00,29.10},
202.   {91.67,118.13},{26.55,51.18},{41.60,74.91},{ 0.39, 6.79},
203.   {86.31,102.08},{20.43,37.80},{ 5.39,28.65},{12.63,24.33},
204.   {22.60,42.79},{ 1.77,14.54},{74.10,113.64},{54.46,87.67},
205.   {18.64,49.32},{93.97,116.30},{42.62,87.04},{13.37,30.16},
206.   {74.50,104.62},{18.28,67.85},{76.98,107.84},{25.89,57.35},
207.   {13.52,42.87},{61.26,97.78},{ 5.97,31.34},{91.99,137.43},
208.   {20.38,58.23},{ 9.59,31.56},{79.41,126.40},{89.90,134.36},
209.   {73.18,111.44},{61.51,111.41},{99.96,147.82},{72.55,113.52},
210.   {66.21,110.93},{36.47,59.41},{65.58,93.39},{24.93,51.71},
211.   {58.00,95.89},{49.83,83.52},{53.35,89.98},{83.97,129.85},
212.   {57.33,106.86},{53.94,98.13},{98.02,144.26},{47.28,72.52},
213.   {45.48,100.70},{80.69,147.66},{96.14,140.01},{82.69,120.80},
```

```
214.   {79.73,136.89},{11.42,27.51},{88.91,138.59},{25.53,51.26},
215.   { 2.49,37.14},{63.89,93.28},{90.96,138.02},{15.27,53.03},
216.   {25.39,51.31},{31.77,55.54},{88.25,124.46},{67.66,108.26},
217.   {90.23,112.02},{17.40,43.85},{78.38,137.07},{96.28,149.45},
218.   {77.38,120.54},{56.49,107.27},{99.00,141.67},{36.35,58.18},
219.   {97.41,132.64},{15.03,48.28},{42.48,81.20},{62.95,105.32},
220.   {99.76,147.11},{85.18,140.95},{99.23,131.84},{21.09,44.44},
221.   {45.12,75.22},{80.36,119.71},{61.37,84.74},{82.64,128.58},
222.   {70.34,108.16},{83.63,116.26},{47.73,67.57},{17.56,48.42},
223.   {23.26,42.12},{41.81,82.17},{18.48,33.63},{39.11,70.14},
224.   {84.20,123.97},{67.20,113.97},{52.74,87.79},{81.66,131.54},
225.   {45.90,93.69},{20.82,34.77},{86.35,122.38},{78.93,106.82},
226.   {10.56,44.66},{51.20,104.61},{93.79,131.97},{15.71,43.06},
227.   {99.16,156.47},{90.70,135.27},{41.85,77.91},{73.41,106.66},
228.   {57.51,108.55},{53.06,115.27},{25.72,67.45},{ 8.03,27.74},
229.   {57.91,101.56},{35.87,57.47},{98.33,145.81},{50.96,76.84},
230.   {57.86,102.10},{17.21,44.21},{95.62,154.59},{76.92,114.77},
231.   {25.32,60.66},{43.60,68.34},{42.68,73.98},{60.36,84.81},
232.   { 9.06,42.91},{ 4.16,18.44},{54.14,97.87},{ 4.87,35.92},
233.   {75.38,112.62},{41.37,68.92},{88.16,163.96},{16.79,41.87},
234.   { 9.77,40.62},{69.66,125.12},{70.35,118.66},{71.99,97.87},
235.   {63.66,111.29},{ 2.01,19.46},{64.63,122.89},{48.39,84.19},
236.   {28.15,64.69},{46.17,83.91},{25.12,45.94},{82.23,118.70},
237.   {57.69,95.98},{24.42,62.91},{15.81,35.58},{75.28,106.87},
238.   {95.74,133.25},{67.78,107.42},{80.89,128.72},{10.39,38.37},
239.   {15.31,35.73},{61.45,110.46},{11.15,44.99},{30.80,63.26},
240.   {84.29,122.39},{29.17,47.34},{80.68,138.44},{81.17,117.86},
241.   { 8.47,32.78},{41.26,74.09},{43.50,71.18},{34.48,68.61},
242.   {30.63,68.05},{88.63,137.28},{71.56,116.97},{21.03,39.12},
243.   {88.20,116.24},{ 8.52,30.24},{95.79,137.27},{78.66,104.62},
244.   {72.44,94.21},{71.60,106.34},{72.11,114.18},{34.50,59.18},
245.   {22.85,60.95},{18.43,40.91},{69.24,119.69},{91.84,142.06},
246.   {34.41,69.95},{95.06,136.92},{67.93,100.93},{46.96,71.82},
247.   {63.92,102.14},{ 1.62,29.66},{95.24,133.60},{43.10,80.88},
248.   {21.83,73.25},{35.01,62.42},{20.05,55.19},{18.64,45.92},
249.   {40.28,75.26},{34.54,63.38},{84.74,117.68},{90.38,144.87},
250.   { 9.91,24.87},{62.97,102.14},{34.40,79.20},{67.34,89.48},
251.   {48.53,85.13},{24.57,51.59},{81.95,117.78},{22.23,49.77},
252.   {75.86,125.20},{60.45,99.78},{19.93,35.57},{48.62,78.46},
253.   {88.49,120.71},{13.33,40.67},{52.03,93.38},{38.43,80.28},
254.   { 2.56,17.00},{18.39,58.10},{58.81,88.08},{75.76,96.69},
255.   {69.78,98.83},{96.47,146.81},{47.32,79.89},{21.90,46.54},
256.   {52.39,83.38},{75.49,107.96},{50.14,80.51},{41.54,73.80},
```

```
257.   {76.07,117.48},{27.00,73.59},{81.59,122.88},{21.74,39.55},
258.   {60.05,105.04},{75.68,102.72},{40.41,79.01},{ 0.32,24.82},
259.   {50.06,106.14},{98.69,139.50},{64.17,109.26},{42.74,78.53},
260.   {39.52,71.78},{55.14,97.37},{25.19,39.08},{99.31,142.63},
261.   {67.50,91.86},{90.92,152.17},{81.99,129.38},{77.28,124.08},
262.   {29.38,69.15},{ 3.81,41.93},{ 9.72,41.83},{25.75,53.09},
263.   {57.28,85.11},{69.50,116.90},{20.00,51.46},{63.00,72.32},
264.   {67.06,102.20},{37.85,64.86},{81.40,114.28},{13.32,58.41},
265.   {67.21,103.77},{63.73,109.66},{91.43,141.66},{54.83,88.07},
266.   {68.03,112.67},{ 0.51,27.76},{ 2.17,38.05},{36.26,66.58},
267.   {72.67,116.52},{98.28,136.37},{85.27,128.64},{90.26,136.47},
268.   {60.31,95.24},{32.77,58.94},{ 3.52,24.75},{15.98,45.49},
269.   {94.25,145.90},{ 8.13,29.89},{61.13,81.38},{44.14,77.64},
270.   {63.53,100.35},{49.35,97.92},{ 4.98,32.12},{25.53,57.45},
271.   { 8.63,41.62},{24.23,56.27},{93.30,137.92},{43.72,71.72},
272.   {54.15,89.12},{ 3.42,36.34},{57.75,85.68},{51.90,87.74},
273.   {85.14,137.82},{99.27,173.87},{82.53,124.94},{15.38,44.42},
274.   {66.66,108.56},{64.12,99.41},{39.08,73.77},{25.42,58.25},
275.   { 1.29,36.39},{98.72,148.84},{70.09,112.06},{ 8.51,27.00},
276.   {85.92,124.74},{88.32,127.04},{51.79,74.58},{36.46,62.45},
277.   {49.29,85.33},{14.06,30.58},{24.83,34.82},{42.85,87.06},
278.   {34.47,76.96},{59.16,90.44},{ 1.02,32.32},{61.80,108.22},
279.   {72.52,95.83},{65.40,99.49},{53.32,93.79},{74.22,117.61},
280.   {53.86,88.31},{39.84,80.11},{79.28,117.86},{34.57,76.73},
281.   {21.69,55.55},{99.87,129.34},{72.12,108.86},{75.08,106.64},
282.   {70.71,106.00},{18.35,67.45},{37.42,66.71},{ 0.70, 9.02},
283.   {56.79,86.75},{74.04,100.45},{53.40,82.23},{42.13,70.45},
284.   {82.43,123.55},{91.65,131.55},{94.99,153.70},{62.14,84.17},
285.   {99.71,151.07},{33.24,73.77},{48.87,76.91},{68.57,118.95},
286.   {14.28,46.22},{18.17,41.01},{95.93,133.32},{ 5.06,33.23},
287.   {57.58,95.47},{18.71,39.10},{90.19,136.73},{26.98,50.08},
288.   {11.36,26.14},{62.70,98.59},{49.32,80.54},{99.97,149.27},
289.   {83.40,132.00},{25.30,48.62},{79.25,117.83},{81.09,109.23},
290.   {31.46,51.02},{14.26,32.26},{33.53,52.63},{ 9.42,47.16},
291.   {67.40,109.90},{18.56,32.79},{34.51,75.14},{49.00,77.38},
292.   {15.69,50.80},{23.09,40.32},{32.03,67.86},{13.60,40.35},
293.   {19.21,60.16},{78.56,111.57},{80.72,131.02},{50.19,79.64},
294.   {55.60,81.78},{ 6.37,43.37},{42.78,74.85},{60.48,113.67},
295.   {44.44,89.27},{54.02,90.24},{73.51,101.74},{16.41,56.73},
296.   {70.94,104.90},{32.03,66.91},{13.12,49.71},{50.16,85.64},
297.   {41.31,68.88},{69.25,123.25},{24.97,69.28},{40.80,86.30},
298.   {32.28,67.01},{90.77,142.80},{66.77,104.70},{24.06,56.12},
299.   {49.16,89.52},{46.10,95.56},{51.79,94.01},{56.11,100.66},
```

```
300.   {88.49,126.71},{ 1.28,21.35},{35.55,64.10},{18.79,29.74},
301.   { 5.40,40.02},{92.32,129.89},{21.13,47.05},{ 5.14,32.16},
302.   {60.89,104.41},{43.45,76.07},{98.91,160.53},{99.31,155.80},
303.   {74.71,121.53},{62.33,98.98},{58.66,101.10},{51.51,93.03},
304.   {51.69,90.42},{19.47,31.22},{85.75,108.87},{64.20,100.48},
305.   {96.60,142.66},{67.99,102.48},{68.37,120.07},{29.81,44.77},
306.   {96.55,142.74},{30.59,43.25},{73.94,108.44},{49.77,88.88},
307.   {59.48,98.21},{41.21,61.86},{38.63,83.41},{86.98,140.40},
308.   {93.34,134.69},{87.92,119.52},{40.93,61.87},{ 2.43,30.68},
309.   {50.74,71.81},{37.13,52.43},{ 1.50,22.18},{99.06,143.48},
310.   { 1.67,27.67},{ 0.18,10.50},{54.13,77.05},{46.19,88.91},
311.   {91.13,144.49},{ 8.95,28.33},{85.69,122.61},{50.30,95.60},
312.   {48.63,103.49},{67.99,100.19},{69.21,112.13},{11.26,34.99},
313.   {25.78,58.73},{84.35,112.36},{46.80,79.68},{69.54,117.99},
314.   {40.30,74.33},{79.97,118.95},{23.28,55.71},{32.62,78.92},
315.   {21.86,37.01},{ 5.07,22.57},{94.41,146.15},{40.14,60.81},
316.   {95.80,125.35},{91.34,131.68},{72.55,113.56},{40.13,71.59},
317.   {98.06,145.27},{90.55,144.08},{71.26,121.81},{33.85,71.13},
318.   {85.74,142.63},{57.93,91.78},{ 7.63,39.30},{83.72,128.26},
319.   {10.89,46.78},{39.79,66.98},{98.84,146.32},{84.62,123.91},
320.   {23.16,31.94},{86.36,134.79},{44.19,63.74},{ 0.39,24.19},
321.   {64.22,96.97},{66.47,103.78},{ 1.73,17.52},{22.25,36.77},
322.   {31.88,59.39},{15.60,30.03},{16.08,41.91},{83.11,129.19},
323.   {72.61,122.52},{19.02,41.06},{56.90,87.53},{65.85,97.02},
324.   {81.40,120.35},{64.90,104.44},{73.35,119.00},{ 8.49,40.31},
325.   {31.20,65.32},{28.29,75.05},{72.51,120.90},{20.42,48.84},
326.   {71.46,111.59},{33.98,50.46},{72.48,111.29},{75.56,113.00},
327.   {58.65,95.16},{23.66,44.95},{95.08,139.46},{80.12,115.20},
328.   {67.77,101.97},{56.06,99.08},{99.03,138.47},{48.26,74.79},
329.   {25.95,39.30},{85.20,137.70},{69.31,104.19},{86.19,122.91},
330.   {37.99,87.47},{72.06,116.90},{ 5.66,28.92},{27.77,52.05},
331.   {31.89,60.32},{18.01,48.92},{37.21,65.49},{73.76,107.20},
332.   { 0.32,-0.71},{93.75,133.48},{69.11,109.63},{11.01,55.84},
333.   {43.48,73.99},{20.76,57.44},{75.50,105.00},{98.74,150.46},
334.   {40.75,90.93},{61.67,103.30},{93.48,155.96},{35.52,61.62},
335.   {32.30,78.52},{28.92,49.61},{60.97,87.11},{13.59,47.58},
336.   { 9.43,26.07},{58.00,107.90},{99.86,151.90},{34.01,57.82},
337.   {39.02,59.14},{33.64,74.99},{ 2.28,20.21},{55.00,90.93},
338.   {55.77,85.94},{79.17,134.03},{63.16,106.70},{17.58,32.28},
339.   {24.29,34.68},{83.91,132.35},{96.44,129.86},{61.95,93.66},
340.   {14.86,25.10},{15.53,33.29},{15.69,42.47},{80.60,126.11},
341.   {16.01,46.33},{26.54,74.55},{ 2.67,37.10},{74.63,96.98},
342.   {38.06,59.99},{56.59,96.87},{78.88,120.95},{87.56,121.75},
```

```
343.   {73.54,119.27},{16.84,44.09},{44.24,89.36},{76.02,123.64},
344.   {98.41,115.45},{12.11,48.19},{30.70,60.41},{55.51,100.49},
345.   { 0.26,37.11},{83.43,124.44},{49.92,111.30},{65.55,99.48},
346.   {77.61,119.44},{62.44,95.52},{21.80,61.06},{20.99,60.54},
347.   {93.10,129.45},{54.96,91.05},{10.22,48.48},{66.77,108.83},
348.   {40.83,87.14},{13.54,35.77},{31.44,62.92},{79.69,110.30},
349.   {67.07,100.59},{28.81,78.71},{52.95,97.30},{39.89,81.67},
350.   {58.79,75.89},{34.35,51.29},{38.03,64.97},{87.87,130.19},
351.   {39.73,52.43},{ 1.64,31.22},{91.15,147.58},{54.08,101.10},
352.   {53.53,74.54},{54.24,104.47},{15.04,51.28},{79.06,114.59},
353.   {93.83,138.37},{94.89,122.18},{52.63,86.22},{27.83,68.05},
354.   {54.51,94.07},{23.83,58.00},{86.88,141.66},{10.42,31.81},
355.   {55.43,84.31},{45.04,85.30},{95.69,121.78},{17.28,35.32},
356.   { 3.17,33.76},{51.61,69.81},{27.37,64.13},{88.92,160.98},
357.   {31.40,64.46},{33.35,59.91},{82.48,128.89},{50.46,98.13},
358.   {78.73,113.68},{70.08,115.27},{98.65,142.28},{ 9.15,50.95},
359.   {16.74,35.73},{32.92,72.02},{ 1.29,18.94},{75.79,123.45},
360.   {32.94,59.92},{61.72,81.50},{42.39,91.90},{70.15,108.81},
361.   { 2.90,29.10},{59.68,87.41},{69.85,108.66},{71.21,107.81},
362.   {24.09,46.47},{44.51,76.59},{ 7.30,34.83},{58.93,99.24},
363.   { 1.24,22.60},{84.27,132.21},{54.11,87.19},{39.18,75.93},
364.   {90.81,155.72},{67.68,88.19},{67.14,84.53},{53.98,86.47},
365.   {67.28,106.68},{ 8.49,36.74},{34.96,62.55},{59.01,82.94},
366.   {64.78,101.77},{66.24,110.82},{75.81,131.28},{62.82,76.02},
367.   {73.95,116.37},{20.40,38.76},{45.06,84.65},{47.64,82.81},
368.   {30.85,64.41},{77.10,112.67},{ 8.12,32.76},{39.56,53.41}
369. };
```

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 |

| 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 |

Write a short analysis of the results.

Linear regression involves finding the equation of the line that best models a set of data points. Comparing the above table we came to know that the time taken for multithreading program take long time than the simple program for linear regression. Here the time taken by the original program 74.7 ns whereas the time taken by the program after implementing the thread is 237707647.8 ns. This is because there are all total 8 threads and only 4 cores to work on. To complete the small work there has been more workload and the threads are spending the time on waiting the other one to complete it.

# 2 CUDA

## 2.1 Password Cracking

```
1. #include <stdio.h>
2. #include <cuda_runtime_api.h>
3. #include <time.h>
4.
5. /******************************************************************************
6.  *
7.  *
8.  * Compile with:
9.  *   nvcc -o cuda PWcuda.cu
10.  *
11.  *
12.  ******************************************************************************/
13.  __device__ int is_a_match(char *attempt){
14.  char password1[] ="PK3467";
15.  char password2[] ="YU7534";
16.  char password3[] ="TH1478";
17.  char password4[] ="LL8970";
18.
19.  char *a = attempt;
20.  char *b = attempt;
21.  char *c = attempt;
22.  char *d = attempt;
23.  char *p1 = password1;
24.  char *p2 = password2;
25.  char *p3 = password3;
```

```
26.  char *p4 = password4;
27.
28.   while(*a ==*p1){
29.          if(*a == '\0')
30.          {
31.                  printf("password:%s\n", password1);
32.                  break;
33.          }
34.          a++;
35.          p1++;
36.  }
37.  while(*b ==*p2){
38.          if(*b == '\0')
39.          {
40.                  printf("password:%s\n", password2);
41.                  break;
42.          }
43.          b++;
44.          p2++;
45.  }
46.  while(*c ==*p3){
47.          if(*c == '\0')
48.          {
49.                  printf("password:%s\n", password3);
50.                  break;
51.          }
52.          c++;
53.          p3++;
54.  }
55.  while(*d ==*p4){
56.          if(*d == '\0')
57.          {
58.                  printf("password: %s\n", password4);
59.                  return 1;
60.          }
61.          d++;
62.          p4++;
63.  }
64.  return 0;
65.  }
66.
67.  __global__ void kernel(){
68.  char i1, i2, i3, i4;
```

```
69.
70.  char password[7];
71.  password[6] ='\0';
72.
73.  int i = blockIdx.x +65;
74.  int j = threadIdx.x+65;
75.  char firstMatch =i;
76.  char secondMatch =j;
77.
78.  password[0] =firstMatch;
79.  password[1] =secondMatch;
80.          for(i1='0'; i1<='9'; i1++){
81.                  for(i2='0'; i2<='9'; i2++){
82.                          for(i3='0'; i3<='9'; i3++){
83.                                  for(i4='0'; i4<='9'; i4++){
84.                                          password[2] =i1;
85.                                          password[3] =i2;
86.                                          password[4] =i3;
87.                                          password[5] =i4;
88.          if(is_a_match(password)){
89.          }
90.          else{
91.          //printf("tried: %s\n",password);
92.          }
93.       }
94.      }
95.    }
96.  }
97.  }
98.
99.  int time_difference(struct timespec *start, struct timespec *finish,long long int *difference) {
100.   long long int ds =  finish->tv_sec - start->tv_sec;
101.   long long int dn =  finish->tv_nsec - start->tv_nsec;
102.
103.   if(dn < 0 ) {
104.     ds--;
105.     dn += 1000000000;
106.   }
107.   *difference = ds * 1000000000 + dn;
108.   return !(*difference > 0);
109. }
110.
111.
```

```
112. int main() {
113.
114.  struct timespec start, finish;
115.  long long int time_elapsed;
116.
117.  clock_gettime(CLOCK_MONOTONIC, &start);
118.
119.   kernel<<<26,26>>>();
120.   cudaThreadSynchronize();
121.
122.
123.  clock_gettime(CLOCK_MONOTONIC, &finish);
124.   time_difference(&start, &finish, &time_elapsed);
125.   printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
126.          (time_elapsed/1.0e9));
127. return 0;
128. }
129.
```

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

| | | | | |
|---|---|---|---|---|
| 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 |

| | | | | |
|---|---|---|---|---|
| Time elapsed was | 67808551 | ns or | 0.67808551 | s |
| Time elapsed was | 57203666 | ns or | 0.57203666 | s |
| Time elapsed was | 56460137 | ns or | 0.56460137 | s |
| Time elapsed was | 58274970 | ns or | 0.5827497 | s |
| Time elapsed was | 59038418 | ns or | 0.59038418 | s |
| Time elapsed was | 56749897 | ns or | 0.56749897 | s |
| Time elapsed was | 64441430 | ns or | 0.6444143 | s |
| Time elapsed was | 59868538 | ns or | 0.59868538 | s |
| Time elapsed was | 67632921 | ns or | 0.67632921 | s |
| Time elapsed was | 61431886 | ns or | 0.61431886 | s |
| mean | 60891041.4 | ns or | 0.60891041 | s |

Write a short analysis of the results

Above table shows the mean running time of both simple and CUDA version of password cracking. Simple program includes 2 alphabet and 2 digit where CUDA program includes 2 alphabet and 4 digit. Compare the table we acknowledge that the mean running time of CUDA program for cracking password is faster than the simple program. This is because, simple program run on the CPU but CUDA program run on GPU which have more number of cores than CPU. GPU compose of several parallel execution units and faster memory interfaces compares to CPU. Thus it is computationally more powerful. This program run 676 threads at a same time in GPU. Mean running time of simple program is 678 sec and CUDA program is 0.60 sec. Hence we can say that CUDA program is 1000 times faster than simple program.

## 2.2 Image Processing

```
1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <time.h>
4. #include <GL/glut.h>
5. #include <GL/gl.h>
6. #include <malloc.h>
7. #include <signal.h>
8. #include <cuda_runtime_api.h>
9.
10.
11.
12.  /*****************************************************************************
13.    Displays two grey scale images. On the left is an image that has come from an
14.    image processing pipeline, just after colour thresholding. On the right is
15.    the result of applying an edge detection convolution operator to the left
16.    image. This program performs that convolution.
17.
18.    Things to note:
19.      - A single unsigned char stores a pixel intensity value. 0 is black, 256 is
20.        white.
21.      - The colour mode used is GL_LUMINANCE. This uses a single number to
22.        represent a pixel's intensity. In this case we want 256 shades of grey,
23.        which is best stored in eight bits, so GL_UNSIGNED_BYTE is specified as
24.        the pixel data type.
25.
26.    To compile adapt the code below wo match your filenames:
27.     nvcc -o Image_processing_cuda Image_processing_cuda.cu -lglut -lGL -lm
28.
29.      To result:
30.          ./Image_processing_cuda
31.
32.  *****************************************************************************/
33.  #define width 100
34.  #define height 72
35.
```

```
36.  unsigned char image[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
37.    0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
38.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
39.    255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
40.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
41.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
42.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
43.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
44.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
45.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
46.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
47.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
48.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
49.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
50.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
51.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
52.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
53.    0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
54.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
55.    255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
56.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
57.    0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
58.    0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,
59.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
60.    255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
61.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
62.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
63.    0,0,0,0,0,0,0,255,255,0,0,0,0,255,255,255,255,255,255,
64.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
65.    255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
66.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
67.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
68.    0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,0,0,255,255,
69.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
70.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
71.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
72.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,0,0,
73.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
74.    255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
75.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
76.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
77.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
78.    255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
79.    0,255,255,0,255,255,255,0,0,255,255,255,255,255,255,255,255,255,255,
80.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
81.    255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
82.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
83.    0,0,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
84.    0,0,0,0,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,
85.    255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
86.    255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
87.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
88.    0,0,0,0,0,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
89.    0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,0,
90.    0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
91.    255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,
92.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
93.    0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,
94.    0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
95.    255,255,255,255,255,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
96.    255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
97.    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
98.    0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
99.    255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
100.   255,255,255,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,255,255,
101.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
102.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
103.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,
104.   255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,255,
105.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
106.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
107.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
108.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
109.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
110.   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
111.   255,255,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
112.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
113.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
114.   0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
115.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
116.   255,255,255,255,255,255,255,255,255,255,0,0,255,255,255,255,255,255,255,255,
117.   255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
118.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
119.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
120.   255,255,255,255,255,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
121.   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,
0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
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,
255,255,0,0,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
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,
255,255,255,255,255,255,255,0,0,255,255,0,255,255,255,255,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
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,
0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
```

```
165.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
166.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
167.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
168.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
169.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
170.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
171.   0,0,255,255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
172.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
173.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
174.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
175.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
176.   0,0,0,0,0,255,255,0,0,0,255,255,255,255,255,255,255,255,255,
177.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
178.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
179.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
180.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
181.   0,0,0,0,0,0,0,0,0,255,0,0,0,0,255,255,255,255,255,
182.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
183.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
184.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
185.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
186.   0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,255,
187.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
188.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,255,255,255,
189.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,
190.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
191.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
192.   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
193.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,255,
194.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
195.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
196.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
197.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
198.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
199.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
200.   255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
201.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
202.   0,0,0,0,0,0,255,0,0,0,0,0,0,0,255,255,255,255,255,
203.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
204.   0,255,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
205.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
206.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
207.   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,
255,255,255,255,255,255,255,255,255,0,0,255,255,0,0,0,0,0,0,0,0,
0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
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,
255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,255,255,0,0,0,
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,
0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
0,255,255,0,0,255,0,255,255,255,255,255,255,255,255,255,255,255,0,0,
0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,0,255,255,255,0,0,0,0,0,255,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
0,0,255,255,0,0,0,255,255,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,0,255,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,255,255,
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,
```

```
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,0,0,255,255,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
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,
255,255,255,255,255,255,255,255,0,255,255,255,0,0,0,0,0,255,
255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,
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,
0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,
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,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
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,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
255,255,0,255,255,255,255,0,255,255,255,255,255,255,255,255,255,255,255,
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,
```

```
294.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
295.   0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,
296.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,255,255,255,255,255,255,
297.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
298.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
299.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
300.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,
301.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
302.   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
303.   0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,0,
304.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
305.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
306.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
307.   255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,
308.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
309.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
310.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
311.   255,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
312.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,
313.   255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
314.   0,0,0,0,0,0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,
315.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
316.   0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
317.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
318.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
319.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
320.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
321.   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,
322.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
323.   255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
324.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
325.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
326.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,
327.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
328.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
329.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
330.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
331.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
332.   0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
333.   255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,
334.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
335.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
336.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,
```

```
337.   0,0,0,255,255,0,0,0,255,255,255,255,255,255,255,255,255,255,255,
338.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,
339.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
340.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
341.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
342.   0,0,255,0,0,0,0,0,0,255,0,0,0,0,255,255,255,255,255,
343.   255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,
344.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
345.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
346.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
347.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,0,0,0,
348.   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
349.   255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
350.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
351.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
352.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
353.   0,0,0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,
354.   255,255,255,255,0,0,0,0,0,0,0,0,255,0,0,0,0,0,0,
355.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
356.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
357.   0,0,0,0,0,0,0,0,0,0,255,255,0,0,255,255,255,255,0,
358.   0,0,0,0,0,0,0,0,0,255,0,0,255,255,255,255,255,255,255,255,
359.   255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,0,0,
360.   0,255,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
361.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
362.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,
363.   255,255,255,255,255,255,0,0,0,0,0,0,0,255,0,0,0,0,255,
364.   255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,
365.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
366.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
367.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,255,255,
368.   255,255,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,0,0,0,
369.   0,0,0,0,0,255,255,255,255,255,255,255,255,255,255,0,0,0,0,0,
370.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
371.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
372.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
373.   0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,0,
374.   0,0,0,0,0,0,0,0,0,0,0,255,255,255,255,255,255,0,0,
375.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
376.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
377.   0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
378.   0,0,0,0,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,
379.   255,255,255,255,255,255,0,0,0,0,0,0,0,0,0,0,255,255,
```

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


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

```
423.
424. __global__ void detect_edges(unsigned char *in, unsigned char *out) {
425.
426.   unsigned int i = (blockIdx.x*72) +threadIdx.x;
427.
428.     int x;//the pixel of interest
429.     int y;//the pixel of interest
430.     int b;//the pixels adjacent to x,y used for calculation
431.     int d;//the pixels adjacent to x,y used for calculation
432.     int f;//the pixels adjacent to x,y used for calculation
433.     int h;//the pixels adjacent to x,y used for calculation
434.     int r;//the result of calculate
435.
436.     y = i / 100;
437.     x = i - (100 * y);
438.
439.     if (x == 0 || y == 0 || x == width - 1 || y == height - 1) {
440.       out[i] = 0;
441.     } else {
442.       b = i + 100;
443.       d = i - 1;
444.       f = i + 1;
445.       h = i - 100;
446.
447.       r = (in[i] * 4) + (in[b] * -1) + (in[d] * -1) + (in[f] * -1) + (in[h] * -1);
448.
449.       if (r > 0) { // if the result is positive this is an edge pixel
450.         out[i] = 255;
451.       } else {
452.         out[i] = 0;
453.       }
454.     }
455.   }
456.
457.
458.
459.
460. void tidy_and_exit() {
461.   exit(0);
462. }
463.
464. void sigint_callback(int signal_number){
465.   printf("\nInterrupt from keyboard\n");
```

```
466.   tidy_and_exit();
467. }
468.
469. static void display() {
470.   glClear(GL_COLOR_BUFFER_BIT);
471.   glRasterPos4i(-1, -1, 0, 1);
472.   glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, image);
473.   glRasterPos4i(0, -1, 0, 1);
474.   glDrawPixels(width, height, GL_LUMINANCE, GL_UNSIGNED_BYTE, results);
475.   glFlush();
476. }
477.
478. static void key_pressed(unsigned char key, int x, int y) {
479.   switch(key){
480.     case 27: // escape
481.       tidy_and_exit();
482.       break;
483.     default:
484.       printf("\nPress escape to exit\n");
485.       break;
486.   }
487. }
488. int time_difference(struct timespec *start, struct timespec *finish,
489.                                 long long int *difference) {
490.   long long int ds =  finish->tv_sec - start->tv_sec;
491.   long long int dn =  finish->tv_nsec - start->tv_nsec;
492.
493.   if(dn < 0 ) {
494.     ds--;
495.     dn += 1000000000;
496.   }
497.   *difference = ds * 1000000000 + dn;
498.   return !(*difference > 0);
499. }
500.
501. int main(int argc, char **argv) {
502.
503.
504.
505.
506.   unsigned char *d_results;
507.   unsigned char *d_image;
508.
```

```
509.
510.   cudaMalloc((void**)&d_image, sizeof(unsigned char) * (width * height) );
511.   cudaMalloc((void**)&d_results, sizeof(unsigned char) * (width * height));
512.   cudaMemcpy(d_image, &image, sizeof(unsigned char) * (width * height), cudaMemcpyHostToDevice);
513.
514.
515. signal(SIGINT, sigint_callback);
516.
517.   printf("image dimensions %dx%d\n", width, height);
518.
519.
520.   struct timespec start, finish;
521.   long long int time_elapsed;
522.
523.   clock_gettime(CLOCK_MONOTONIC, &start);
524.   printf("image dimensions %dx%d\n", width, height);
525.   detect_edges <<<100, 72>>>(d_image, d_results);
526.   cudaThreadSynchronize();
527.   cudaMemcpy(&results, d_results, sizeof(unsigned char) * (width * height), cudaMemcpyDeviceToHost);
528.   clock_gettime(CLOCK_MONOTONIC, &finish);
529.
530.   time_difference(&start, &finish, &time_elapsed);
531.   printf("Time elapsed was %lldns or %0.9lfs\n",
532.   time_elapsed, (time_elapsed/1.0e9));
533.
534.
535.
536.   cudaFree(&d_image);
537.   cudaFree(&d_results);
538.
539.
540.
541.   glutInit(&argc, argv);
542.   glutInitWindowSize(width * 2,height);
543.   glutInitDisplayMode(GLUT_SINGLE | GLUT_LUMINANCE);
544.
545.   glutCreateWindow("6CS005 Image Progessing Courework");
546.   glutDisplayFunc(display);
547.   glutKeyboardFunc(key_pressed);
548.   glClearColor(0.0, 1.0, 0.0, 1.0);
549.
550.
551.   glutMainLoop();
```

```
552.    tidy_and_exit();
553.
554.
555.    return 0;
556. }
```

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

| | | | |
|---|---|---|---|
| Time elapsed was | 83 | ns or | 0.000000083 |
| Time elapsed was | 129 | ns or | 0.000000129 |
| Time elapsed was | 109 | ns or | 0.000000109 |
| Time elapsed was | 79 | ns or | 0.000000079 |
| Time elapsed was | 68 | ns or | 0.000000068 |
| Time elapsed was | 101 | ns or | 0.000000101 |
| Time elapsed was | 83 | ns or | 0.000000083 |
| Time elapsed was | 126 | ns or | 0.000000126 |
| Time elapsed was | 78 | ns or | 0.000000078 |
| Time elapsed was | 81 | ns or | 0.000000081 |
| mean | 93.7 | | 0.0000000937 |

| | | | | |
|---|---|---|---|---|
| Time elapsed was | 42250 | ns or | 0.00004225 | s |
| Time elapsed was | 52876 | ns or | 0.00005288 | s |
| Time elapsed was | 50446 | ns or | 0.00005045 | s |
| Time elapsed was | 50875 | ns or | 0.00005088 | s |
| Time elapsed was | 56083 | ns or | 0.00005608 | s |
| Time elapsed was | 41803 | ns or | 0.0000418 | s |
| Time elapsed was | 43474 | ns or | 0.00004347 | s |
| Time elapsed was | 47480 | ns or | 0.00004748 | s |
| Time elapsed was | 42784 | ns or | 0.00004278 | s |
| Time elapsed was | 52052 | ns or | 0.00005205 | s |
| Average mean Time | 48012.3 | ns or | 0.000048012 | s |

Write a short analysis of the results

The table above shows the mean running time of simple program and CUDA program for image processing. The mean time for the original version was 93.7ns whereas the mean time for the CUDA program is 48012.3 ns. The difference between them was 47918.6 ns. In this case we find that programmed executed in GPU is slower than the one that is runs in CPU. This happens because the data of image should be copped form host to device and in device the program is executed. After executing device should again pass the data to host for displaying. This process make CUDA program slower than original program.

## 2.3 Linear Regression

```
1. #include <stdio.h>
2. #include <math.h>
3. #include <time.h>
4. #include <unistd.h>
5. #include <cuda_runtime_api.h>
6. #include <errno.h>
7. #include <unistd.h>
8. /******************************************************************************
9.  * This program takes an initial estimate of m and c and finds the associated
10.  * rms error. It is then as a base to generate and evaluate 8 new estimates,
11.  * which are steps in different directions in m-c space. The best estimate is
12.  * then used as the base for another iteration of "generate and evaluate". This
13.  * continues until none of the new estimates are better than the base. This is
14.  * a gradient search for a minimum in mc-space.
15.  *
```

```
 * To compile:
 *   nvcc -o linearcuda linear_cuda.cu -lm
 *
 * To run:
 *   ./linearcuda
 *
 *
 ****************************************************************************/

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

int n_data = 1000;
__device__ int d_n_data =1000;

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},
```

```
59.    {90.47,126.33},{26.80,44.96},{75.94,119.76},{33.83,69.11},
60.    {63.59,103.98},{38.05,72.36},{68.28,110.76},{ 3.34,54.22},
61.    {45.40,92.84},{78.37,113.49},{27.11,46.46},{32.32,68.44},
62.    {20.97,30.90},{37.92,75.11},{96.85,130.96},{69.40,95.17},
63.    { 3.29,30.06},{64.41,103.44},{15.80,52.64},{61.76,97.79},
64.    { 1.62,33.98},{29.03,58.02},{18.74,34.93},{25.41,73.73},
65.    {28.78,65.94},{14.64,50.31},{82.85,133.70},{41.62,90.32},
66.    {99.28,144.95},{90.16,133.18},{40.45,77.72},{ 1.79,50.44},
67.    {31.80,62.71},{26.30,40.89},{47.57,83.15},{17.78,44.90},
68.    {69.48,93.13},{87.98,126.95},{69.84,106.00},{37.06,61.61},
69.    {90.65,133.97},{10.73,46.60},{38.84,79.90},{ 4.75,33.89},
70.    {48.99,89.31},{ 2.51,47.09},{34.99,86.40},{29.79,54.52},
71.    {91.30,133.72},{74.12,122.86},{90.93,141.88},{51.14,89.93},
72.    {84.53,142.49},{26.84,58.79},{ 6.95,20.98},{49.80,85.14},
73.    {22.82,57.02},{44.08,89.32},{22.28,48.72},{21.12,50.68},
74.    {65.69,93.93},{27.84,39.97},{ 1.92,40.39},{ 9.36,33.54},
75.    {88.10,123.02},{18.15,63.84},{21.80,39.76},{64.42,101.03},
76.    { 2.23,22.52},{55.68,99.56},{37.55,87.77},{74.23,104.87},
77.    {11.96,37.30},{23.60,45.84},{11.13,34.32},{ 9.05,48.79},
78.    {56.11,100.21},{19.31,54.44},{ 6.27,16.17},{64.65,101.39},
79.    {50.25,77.59},{69.33,95.12},{47.52,87.79},{28.97,65.98},
80.    {71.56,95.30},{19.71,41.47},{57.66,96.65},{41.07,74.10},
81.    {35.08,79.46},{40.80,87.01},{ 0.31,19.82},{90.78,111.55},
82.    {34.39,72.03},{99.97,139.40},{30.86,73.03},{14.37,50.15},
83.    { 6.11,42.76},{21.75,80.30},{89.94,127.56},{10.86,42.40},
84.    {13.07,42.98},{84.47,147.14},{83.44,132.18},{32.24,63.57},
85.    {66.93,102.41},{34.48,68.96},{ 3.46,22.82},{94.84,130.83},
86.    {49.41,107.26},{71.64,99.82},{47.28,80.62},{39.17,68.77},
87.    {58.05,108.35},{69.27,109.81},{47.64,73.34},{34.64,73.15},
88.    {22.86,46.34},{37.76,66.19},{ 3.12,39.11},{60.59,111.05},
89.    {91.99,122.76},{96.60,138.86},{ 3.58,23.35},{22.81,60.18},
90.    {13.93,21.32},{69.51,106.41},{19.57,43.39},{79.11,115.68},
91.    {80.89,124.36},{44.42,57.78},{33.28,73.04},{21.45,49.88},
92.    {70.57,113.77},{45.63,65.60},{55.99,72.21},{21.62,41.47},
93.    {61.74,98.99},{ 9.30,29.77},{75.32,106.74},{27.97,73.44},
94.    {74.77,115.98},{42.93,82.67},{92.32,138.05},{25.55,64.34},
95.    { 0.48,23.51},{79.52,111.52},{52.83,70.58},{51.45,87.28},
96.    {62.72,90.41},{ 4.16,40.60},{70.13,115.25},{55.96,97.34},
97.    {93.88,154.09},{46.21,90.04},{34.75,51.46},{54.45,89.56},
98.    {80.69,129.36},{45.14,73.00},{47.34,85.69},{70.16,118.02},
99.    { 4.26,17.14},{61.56,98.04},{15.95,28.56},{74.06,118.48},
100.   {65.29,99.71},{19.08,55.64},{37.82,72.36},{58.22,103.93},
101.   {50.52,82.15},{26.25,60.91},{97.77,123.91},{39.13,68.03},
```

```
102.   {15.09,41.88},{32.61,61.64},{11.23,22.85},{61.92,98.02},
103.   {73.63,126.32},{35.12,54.74},{12.98,42.69},{83.87,128.60},
104.   {45.65,78.81},{42.85,90.57},{76.74,117.53},{19.05,49.60},
105.   {69.03,104.16},{23.66,54.97},{52.85,85.94},{82.07,128.27},
106.   {74.77,111.22},{95.04,136.69},{40.49,49.53},{ 4.16,28.40},
107.   { 7.69,51.29},{29.37,80.82},{86.06,122.19},{ 3.92,23.24},
108.   {62.76,108.89},{27.12,54.24},{10.24,33.84},{79.86,107.97},
109.   {57.09,85.27},{10.29,54.38},{53.50,82.98},{12.83,50.29},
110.   { 2.09,13.69},{88.73,135.16},{42.72,87.10},{40.20,91.88},
111.   {40.10,76.49},{80.22,133.65},{57.55,93.99},{29.34,69.08},
112.   { 2.90,41.26},{44.60,82.03},{47.93,89.05},{98.17,123.11},
113.   {17.21,45.91},{42.37,79.83},{90.89,119.42},{ 7.81,36.64},
114.   {76.14,123.86},{47.79,83.40},{95.27,144.30},{44.13,98.20},
115.   {19.97,37.36},{90.66,131.96},{75.41,117.80},{57.14,107.91},
116.   {25.92,41.69},{90.86,130.36},{44.78,79.02},{23.00,29.10},
117.   {91.67,118.13},{26.55,51.18},{41.60,74.91},{ 0.39, 6.79},
118.   {86.31,102.08},{20.43,37.80},{ 5.39,28.65},{12.63,24.33},
119.   {22.60,42.79},{ 1.77,14.54},{74.10,113.64},{54.46,87.67},
120.   {18.64,49.32},{93.97,116.30},{42.62,87.04},{13.37,30.16},
121.   {74.50,104.62},{18.28,67.85},{76.98,107.84},{25.89,57.35},
122.   {13.52,42.87},{61.26,97.78},{ 5.97,31.34},{91.99,137.43},
123.   {20.38,58.23},{ 9.59,31.56},{79.41,126.40},{89.90,134.36},
124.   {73.18,111.44},{61.51,111.41},{99.96,147.82},{72.55,113.52},
125.   {66.21,110.93},{36.47,59.41},{65.58,93.39},{24.93,51.71},
126.   {58.00,95.89},{49.83,83.52},{53.35,89.98},{83.97,129.85},
127.   {57.33,106.86},{53.94,98.13},{98.02,144.26},{47.28,72.52},
128.   {45.48,100.70},{80.69,147.66},{96.14,140.01},{82.69,120.80},
129.   {79.73,136.89},{11.42,27.51},{88.91,138.59},{25.53,51.26},
130.   { 2.49,37.14},{63.89,93.28},{90.96,138.02},{15.27,53.03},
131.   {25.39,51.31},{31.77,55.54},{88.25,124.46},{67.66,108.26},
132.   {90.23,112.02},{17.40,43.85},{78.38,137.07},{96.28,149.45},
133.   {77.38,120.54},{56.49,107.27},{99.00,141.67},{36.35,58.18},
134.   {97.41,132.64},{15.03,48.28},{42.48,81.20},{62.95,105.32},
135.   {99.76,147.11},{85.18,140.95},{99.23,131.84},{21.09,44.44},
136.   {45.12,75.22},{80.36,119.71},{61.37,84.74},{82.64,128.58},
137.   {70.34,108.16},{83.63,116.26},{47.73,67.57},{17.56,48.42},
138.   {23.26,42.12},{41.81,82.17},{18.48,33.63},{39.11,70.14},
139.   {84.20,123.97},{67.20,113.97},{52.74,87.79},{81.66,131.54},
140.   {45.90,93.69},{20.82,34.77},{86.35,122.38},{78.93,106.82},
141.   {10.56,44.66},{51.20,104.61},{93.79,131.97},{15.71,43.06},
142.   {99.16,156.47},{90.70,135.27},{41.85,77.91},{73.41,106.66},
143.   {57.51,108.55},{53.06,115.27},{25.72,67.45},{ 8.03,27.74},
144.   {57.91,101.56},{35.87,57.47},{98.33,145.81},{50.96,76.84},
```

```
145.   {57.86,102.10},{17.21,44.21},{95.62,154.59},{76.92,114.77},
146.   {25.32,60.66},{43.60,68.34},{42.68,73.98},{60.36,84.81},
147.   { 9.06,42.91},{ 4.16,18.44},{54.14,97.87},{ 4.87,35.92},
148.   {75.38,112.62},{41.37,68.92},{88.16,163.96},{16.79,41.87},
149.   { 9.77,40.62},{69.66,125.12},{70.35,118.66},{71.99,97.87},
150.   {63.66,111.29},{ 2.01,19.46},{64.63,122.89},{48.39,84.19},
151.   {28.15,64.69},{46.17,83.91},{25.12,45.94},{82.23,118.70},
152.   {57.69,95.98},{24.42,62.91},{15.81,35.58},{75.28,106.87},
153.   {95.74,133.25},{67.78,107.42},{80.89,128.72},{10.39,38.37},
154.   {15.31,35.73},{61.45,110.46},{11.15,44.99},{30.80,63.26},
155.   {84.29,122.39},{29.17,47.34},{80.68,138.44},{81.17,117.86},
156.   { 8.47,32.78},{41.26,74.09},{43.50,71.18},{34.48,68.61},
157.   {30.63,68.05},{88.63,137.28},{71.56,116.97},{21.03,39.12},
158.   {88.20,116.24},{ 8.52,30.24},{95.79,137.27},{78.66,104.62},
159.   {72.44,94.21},{71.60,106.34},{72.11,114.18},{34.50,59.18},
160.   {22.85,60.95},{18.43,40.91},{69.24,119.69},{91.84,142.06},
161.   {34.41,69.95},{95.06,136.92},{67.93,100.93},{46.96,71.82},
162.   {63.92,102.14},{ 1.62,29.66},{95.24,133.60},{43.10,80.88},
163.   {21.83,73.25},{35.01,62.42},{20.05,55.19},{18.64,45.92},
164.   {40.28,75.26},{34.54,63.38},{84.74,117.68},{90.38,144.87},
165.   { 9.91,24.87},{62.97,102.14},{34.40,79.20},{67.34,89.48},
166.   {48.53,85.13},{24.57,51.59},{81.95,117.78},{22.23,49.77},
167.   {75.86,125.20},{60.45,99.78},{19.93,35.57},{48.62,78.46},
168.   {88.49,120.71},{13.33,40.67},{52.03,93.38},{38.43,80.28},
169.   { 2.56,17.00},{18.39,58.10},{58.81,88.08},{75.76,96.69},
170.   {69.78,98.83},{96.47,146.81},{47.32,79.89},{21.90,46.54},
171.   {52.39,83.38},{75.49,107.96},{50.14,80.51},{41.54,73.80},
172.   {76.07,117.48},{27.00,73.59},{81.59,122.88},{21.74,39.55},
173.   {60.05,105.04},{75.68,102.72},{40.41,79.01},{ 0.32,24.82},
174.   {50.06,106.14},{98.69,139.50},{64.17,109.26},{42.74,78.53},
175.   {39.52,71.78},{55.14,97.37},{25.19,39.08},{99.31,142.63},
176.   {67.50,91.86},{90.92,152.17},{81.99,129.38},{77.28,124.08},
177.   {29.38,69.15},{ 3.81,41.93},{ 9.72,41.83},{25.75,53.09},
178.   {57.28,85.11},{69.50,116.90},{20.00,51.46},{63.00,72.32},
179.   {67.06,102.20},{37.85,64.86},{81.40,114.28},{13.32,58.41},
180.   {67.21,103.77},{63.73,109.66},{91.43,141.66},{54.83,88.07},
181.   {68.03,112.67},{ 0.51,27.76},{ 2.17,38.05},{36.26,66.58},
182.   {72.67,116.52},{98.28,136.37},{85.27,128.64},{90.26,136.47},
183.   {60.31,95.24},{32.77,58.94},{ 3.52,24.75},{15.98,45.49},
184.   {94.25,145.90},{ 8.13,29.89},{61.13,81.38},{44.14,77.64},
185.   {63.53,100.35},{49.35,97.92},{ 4.98,32.12},{25.53,57.45},
186.   { 8.63,41.62},{24.23,56.27},{93.30,137.92},{43.72,71.72},
187.   {54.15,89.12},{ 3.42,36.34},{57.75,85.68},{51.90,87.74},
```

```
188.   {85.14,137.82},{99.27,173.87},{82.53,124.94},{15.38,44.42},
189.   {66.66,108.56},{64.12,99.41},{39.08,73.77},{25.42,58.25},
190.   { 1.29,36.39},{98.72,148.84},{70.09,112.06},{ 8.51,27.00},
191.   {85.92,124.74},{88.32,127.04},{51.79,74.58},{36.46,62.45},
192.   {49.29,85.33},{14.06,30.58},{24.83,34.82},{42.85,87.06},
193.   {34.47,76.96},{59.16,90.44},{ 1.02,32.32},{61.80,108.22},
194.   {72.52,95.83},{65.40,99.49},{53.32,93.79},{74.22,117.61},
195.   {53.86,88.31},{39.84,80.11},{79.28,117.86},{34.57,76.73},
196.   {21.69,55.55},{99.87,129.34},{72.12,108.86},{75.08,106.64},
197.   {70.71,106.00},{18.35,67.45},{37.42,66.71},{ 0.70, 9.02},
198.   {56.79,86.75},{74.04,100.45},{53.40,82.23},{42.13,70.45},
199.   {82.43,123.55},{91.65,131.55},{94.99,153.70},{62.14,84.17},
200.   {99.71,151.07},{33.24,73.77},{48.87,76.91},{68.57,118.95},
201.   {14.28,46.22},{18.17,41.01},{95.93,133.32},{ 5.06,33.23},
202.   {57.58,95.47},{18.71,39.10},{90.19,136.73},{26.98,50.08},
203.   {11.36,26.14},{62.70,98.59},{49.32,80.54},{99.97,149.27},
204.   {83.40,132.00},{25.30,48.62},{79.25,117.83},{81.09,109.23},
205.   {31.46,51.02},{14.26,32.26},{33.53,52.63},{ 9.42,47.16},
206.   {67.40,109.90},{18.56,32.79},{34.51,75.14},{49.00,77.38},
207.   {15.69,50.80},{23.09,40.32},{32.03,67.86},{13.60,40.35},
208.   {19.21,60.16},{78.56,111.57},{80.72,131.02},{50.19,79.64},
209.   {55.60,81.78},{ 6.37,43.37},{42.78,74.85},{60.48,113.67},
210.   {44.44,89.27},{54.02,90.24},{73.51,101.74},{16.41,56.73},
211.   {70.94,104.90},{32.03,66.91},{13.12,49.71},{50.16,85.64},
212.   {41.31,68.88},{69.25,123.25},{24.97,69.28},{40.80,86.30},
213.   {32.28,67.01},{90.77,142.80},{66.77,104.70},{24.06,56.12},
214.   {49.16,89.52},{46.10,95.56},{51.79,94.01},{56.11,100.66},
215.   {88.49,126.71},{ 1.28,21.35},{35.55,64.10},{18.79,29.74},
216.   { 5.40,40.02},{92.32,129.89},{21.13,47.05},{ 5.14,32.16},
217.   {60.89,104.41},{43.45,76.07},{98.91,160.53},{99.31,155.80},
218.   {74.71,121.53},{62.33,98.98},{58.66,101.10},{51.51,93.03},
219.   {51.69,90.42},{19.47,31.22},{85.75,108.87},{64.20,100.48},
220.   {96.60,142.66},{67.99,102.48},{68.37,120.07},{29.81,44.77},
221.   {96.55,142.74},{30.59,43.25},{73.94,108.44},{49.77,88.88},
222.   {59.48,98.21},{41.21,61.86},{38.63,83.41},{86.98,140.40},
223.   {93.34,134.69},{87.92,119.52},{40.93,61.87},{ 2.43,30.68},
224.   {50.74,71.81},{37.13,52.43},{ 1.50,22.18},{99.06,143.48},
225.   { 1.67,27.67},{ 0.18,10.50},{54.13,77.05},{46.19,88.91},
226.   {91.13,144.49},{ 8.95,28.33},{85.69,122.61},{50.30,95.60},
227.   {48.63,103.49},{67.99,100.19},{69.21,112.13},{11.26,34.99},
228.   {25.78,58.73},{84.35,112.36},{46.80,79.68},{69.54,117.99},
229.   {40.30,74.33},{79.97,118.95},{23.28,55.71},{32.62,78.92},
230.   {21.86,37.01},{ 5.07,22.57},{94.41,146.15},{40.14,60.81},
```

```
231.   {95.80,125.35},{91.34,131.68},{72.55,113.56},{40.13,71.59},
232.   {98.06,145.27},{90.55,144.08},{71.26,121.81},{33.85,71.13},
233.   {85.74,142.63},{57.93,91.78},{ 7.63,39.30},{83.72,128.26},
234.   {10.89,46.78},{39.79,66.98},{98.84,146.32},{84.62,123.91},
235.   {23.16,31.94},{86.36,134.79},{44.19,63.74},{ 0.39,24.19},
236.   {64.22,96.97},{66.47,103.78},{ 1.73,17.52},{22.25,36.77},
237.   {31.88,59.39},{15.60,30.03},{16.08,41.91},{83.11,129.19},
238.   {72.61,122.52},{19.02,41.06},{56.90,87.53},{65.85,97.02},
239.   {81.40,120.35},{64.90,104.44},{73.35,119.00},{ 8.49,40.31},
240.   {31.20,65.32},{28.29,75.05},{72.51,120.90},{20.42,48.84},
241.   {71.46,111.59},{33.98,50.46},{72.48,111.29},{75.56,113.00},
242.   {58.65,95.16},{23.66,44.95},{95.08,139.46},{80.12,115.20},
243.   {67.77,101.97},{56.06,99.08},{99.03,138.47},{48.26,74.79},
244.   {25.95,39.30},{85.20,137.70},{69.31,104.19},{86.19,122.91},
245.   {37.99,87.47},{72.06,116.90},{ 5.66,28.92},{27.77,52.05},
246.   {31.89,60.32},{18.01,48.92},{37.21,65.49},{73.76,107.20},
247.   { 0.32,-0.71},{93.75,133.48},{69.11,109.63},{11.01,55.84},
248.   {43.48,73.99},{20.76,57.44},{75.50,105.00},{98.74,150.46},
249.   {40.75,90.93},{61.67,103.30},{93.48,155.96},{35.52,61.62},
250.   {32.30,78.52},{28.92,49.61},{60.97,87.11},{13.59,47.58},
251.   { 9.43,26.07},{58.00,107.90},{99.86,151.90},{34.01,57.82},
252.   {39.02,59.14},{33.64,74.99},{ 2.28,20.21},{55.00,90.93},
253.   {55.77,85.94},{79.17,134.03},{63.16,106.70},{17.58,32.28},
254.   {24.29,34.68},{83.91,132.35},{96.44,129.86},{61.95,93.66},
255.   {14.86,25.10},{15.53,33.29},{15.69,42.47},{80.60,126.11},
256.   {16.01,46.33},{26.54,74.55},{ 2.67,37.10},{74.63,96.98},
257.   {38.06,59.99},{56.59,96.87},{78.88,120.95},{87.56,121.75},
258.   {73.54,119.27},{16.84,44.09},{44.24,89.36},{76.02,123.64},
259.   {98.41,115.45},{12.11,48.19},{30.70,60.41},{55.51,100.49},
260.   { 0.26,37.11},{83.43,124.44},{49.92,111.30},{65.55,99.48},
261.   {77.61,119.44},{62.44,95.52},{21.80,61.06},{20.99,60.54},
262.   {93.10,129.45},{54.96,91.05},{10.22,48.48},{66.77,108.83},
263.   {40.83,87.14},{13.54,35.77},{31.44,62.92},{79.69,110.30},
264.   {67.07,100.59},{28.81,78.71},{52.95,97.30},{39.89,81.67},
265.   {58.79,75.89},{34.35,51.29},{38.03,64.97},{87.87,130.19},
266.   {39.73,52.43},{ 1.64,31.22},{91.15,147.58},{54.08,101.10},
267.   {53.53,74.54},{54.24,104.47},{15.04,51.28},{79.06,114.59},
268.   {93.83,138.37},{94.89,122.18},{52.63,86.22},{27.83,68.05},
269.   {54.51,94.07},{23.83,58.00},{86.88,141.66},{10.42,31.81},
270.   {55.43,84.31},{45.04,85.30},{95.69,121.78},{17.28,35.32},
271.   { 3.17,33.76},{51.61,69.81},{27.37,64.13},{88.92,160.98},
272.   {31.40,64.46},{33.35,59.91},{82.48,128.89},{50.46,98.13},
273.   {78.73,113.68},{70.08,115.27},{98.65,142.28},{ 9.15,50.95},
```

```
274.   {16.74,35.73},{32.92,72.02},{ 1.29,18.94},{75.79,123.45},
275.   {32.94,59.92},{61.72,81.50},{42.39,91.90},{70.15,108.81},
276.   { 2.90,29.10},{59.68,87.41},{69.85,108.66},{71.21,107.81},
277.   {24.09,46.47},{44.51,76.59},{ 7.30,34.83},{58.93,99.24},
278.   { 1.24,22.60},{84.27,132.21},{54.11,87.19},{39.18,75.93},
279.   {90.81,155.72},{67.68,88.19},{67.14,84.53},{53.98,86.47},
280.   {67.28,106.68},{ 8.49,36.74},{34.96,62.55},{59.01,82.94},
281.   {64.78,101.77},{66.24,110.82},{75.81,131.28},{62.82,76.02},
282.   {73.95,116.37},{20.40,38.76},{45.06,84.65},{47.64,82.81},
283.   {30.85,64.41},{77.10,112.67},{ 8.12,32.76},{39.56,53.41}
284. };
285. double residual_error(double x, double y, double m, double c) {
286.   double e = (m * x) + c - y;
287.   return e * e;
288. }
289. __device__ double d_residual_error(double x, double y, double m, double c) {
290.   double e = (m * x) + c - y;
291.   return e * e;
292. }
293. double rms_error(double m, double c) {
294.   int i;
295.   double mean;
296.   double error_sum = 0;
297.
298.   for(i=0; i<n_data; i++) {
299.     error_sum += residual_error(data[i].x, data[i].y, m, c);
300.   }
301.
302.   mean = error_sum / n_data;
303.
304.   return sqrt(mean);
305. }
306. __global__ void d_rms_error(double *m, double *c,double *error_sum_arr,point_t *d_data) {
307.   int i = threadIdx.x + blockIdx.x *blockDim.x;
308.         error_sum_arr[i] = d_residual_error(d_data[i].x,d_data[i].y, *m, *c);
309.         }
310.
311. int time_difference(struct timespec *start, struct timespec *finish, long long int *difference)
312.         {
313.                 long long int ds = finish->tv_sec - start->tv_sec;
314.                 long long int dn = finish->tv_nsec - start->tv_nsec;
315.
316.                 if(dn < 0){
```

```
317.                 ds--;
318.                 dn += 1000000000;
319.         }
320.                 *difference = ds * 1000000000 + dn;
321.                 return !(*difference > 0);
322. }
323.
324.
325.
326. int main(){
327.  int i;
328.   double bm = 1.3;
329.   double bc = 10;
330.   double be;
331.   double dm[8];
332.   double dc[8];
333.   double e[8];
334.   double step = 0.01;
335.   double best_error = 999999999;
336.   int best_error_i;
337.   int minimum_found = 0;
338.
339.   double om[] = {0,1,1, 1, 0,-1,-1,-1};
340.   double oc[] = {1,1,0,-1,-1,-1, 0, 1};
341.
342. struct timespec start, finish;
343.         long long int time_elapsed;
344.         clock_gettime(CLOCK_MONOTONIC, &start);
345.         cudaError_t error;
346.
347.
348. double *d_dm;
349. double *d_dc;
350. double *d_error_sum_arr;
351. point_t *d_data;
352.
353. be= rms_error(bm,bc);
354.
355. error=cudaMalloc(&d_dm,(sizeof(double) * 8));
356.         if(error){
357.                 fprintf(stderr,"cudaMalloc on d_dm returned %d %s\n",error,
358.                 cudaGetErrorString(error));
359.                 exit(1);
```

```
360. }
361. error=cudaMalloc(&d_dc,(sizeof(double) * 8));
362. if(error){
363.         fprintf(stderr,"cudaMalloc on d_dc returned %d %s\n",error,
364.         cudaGetErrorString(error));
365.         exit(1);
366. }
367.
368. error=cudaMalloc(&d_error_sum_arr,(sizeof(double) * 1000));
369. if(error){
370.         fprintf(stderr,"cudaMalloc on d_error_sum_arr returned %d %s\n",error, //371
371.         cudaGetErrorString(error));
372.         exit(1);
373. }
374.
375. error=cudaMalloc(&d_data,sizeof(data)); //376
376. if(error){
377.         fprintf(stderr,"cudaMalloc on d_data returned %d %s\n",error,
378.         cudaGetErrorString(error));
379.         exit(1);
380. }
381.
382. while(!minimum_found) {
383.     for(i=0;i<8;i++) {
384. dm[i] = bm + (om[i] * step);
385. dc[i]= bc + (oc[i] * step);
386. }
387.
388.  error = cudaMemcpy(d_dm,dm,(sizeof(double)*8), cudaMemcpyHostToDevice);
389. if(error){
390.         fprintf(stderr,"cudaMemcpy to d_dm returned %d %s\n",error,
391.         cudaGetErrorString(error));
392. }
393.
394.  error = cudaMemcpy(d_dc,dc,(sizeof(double)*8), cudaMemcpyHostToDevice);
395. if(error){
396.         fprintf(stderr,"cudaMemcpy to d_dc returned %d %s\n",error,
397.         cudaGetErrorString(error));
398. }
399.
400. error = cudaMemcpy(d_data, data,sizeof(data), cudaMemcpyHostToDevice); //401
401. if(error){
402.         fprintf(stderr,"cudaMemcpy to d_data returned %d %s\n",error,
```

```
403.         cudaGetErrorString(error));
404. }
405.
406. for(i=0;i<8;i++){
407. double h_error_sum_arr[1000];
408.
409. double error_sum_total;
410. double error_sum_mean;
411.
412. d_rms_error <<<100,10>>>(&d_dm[i],&d_dc[i],d_error_sum_arr,d_data);
413.         cudaThreadSynchronize();
414.         error =cudaMemcpy(&h_error_sum_arr,d_error_sum_arr,(sizeof(double) *1000),
415.         cudaMemcpyDeviceToHost);
416. if(error){
417.         fprintf(stderr,"cudaMemcpy to error_sum returned %d %s\n",error,
418.         cudaGetErrorString(error));
419. }
420. for(int j=0;j<n_data;j++){
421.         error_sum_total+= h_error_sum_arr[j];
422. }
423.         error_sum_mean = error_sum_total / n_data;
424.         e[i] =sqrt(error_sum_mean);
425.
426. if(e[i] < best_error){
427.         best_error = e[i];
428.         error_sum_total +=h_error_sum_arr[i];
429. }
430. error_sum_mean = error_sum_total /n_data;//431
431. e[i] =  sqrt(error_sum_mean); //432
432.
433. if(e[i]<best_error){ //434
434.         best_error = e[i];
435.         best_error_i = i;
436. }
437.  error_sum_total = 0;  //438
438. }
439. if(best_error <be){
440. be=best_error;
441. bm =dm[best_error_i];
442. bc= dc[best_error_i];
443. }else {
444. minimum_found = 1;
445. }
```

```
446. }
447.
448.
449. error = cudaFree(d_dm);
450. if(error){
451. fprintf(stderr,"cudaFree on d_dm returned %d %s\n",error,
452. cudaGetErrorString(error));  //453
453. exit(1);
454. }
455.
456. error = cudaFree(d_dc);
457. if(error){
458. fprintf(stderr,"cudaFree on d_dc returned %d %s\n",error,
459. cudaGetErrorString(error));
460. exit(1);
461. }
462.
463. error = cudaFree(d_data);
464. if(error){
465. fprintf(stderr,"cudaFree on d_data returned %d %s\n",error,
466. cudaGetErrorString(error));
467. exit(1);
468. }
469.
470. error = cudaFree(d_error_sum_arr);
471. if(error){
472. fprintf(stderr,"cudaFree on d_error_sum_arr returned %d %s\n",error,
473. cudaGetErrorString(error));
474. exit(1);
475. }
476.
477.
478. printf("minimum m,c is %lf,%lf with error %lf\n", bm, bc, be);
479.
480. clock_gettime(CLOCK_MONOTONIC, &finish);
481.   time_difference(&start, &finish, &time_elapsed);
482.   printf("Time elapsed was %lldns or %0.9lfs\n", time_elapsed,
483.                                          (time_elapsed/1.0e9));
484.
485. return 0;
486. }
487.
488. ;
```

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

| | | | | |
|---|---|---|---|---|
| Time elapsed was | 82 | ns or | 0.000000082 | s |
| Time elapsed was | 110 | ns or | 0.000000110 | s |
| Time elapsed was | 61 | ns or | 0.000000061 | s |
| Time elapsed was | 68 | ns or | 0.000000068 | s |
| Time elapsed was | 83 | ns or | 0.000000083 | s |
| Time elapsed was | 81 | ns or | 0.000000081 | s |
| Time elapsed was | 77 | ns or | 0.000000077 | s |
| Time elapsed was | 57 | ns or | 0.000000057 | s |
| Time elapsed was | 67 | ns or | 0.000000067 | s |
| Time elapsed was | 61 | ns or | 0.000000061 | s |
| total | 747 | ns or | 0.000000747 | s |
| Mean | 74.7 | ns or | 0.000000075 | s |
| Average Mean | 37.35 | ns or | 37.35 | s |

| | | | | |
|---|---|---|---|---|
| Time elapsed was | 280413117 | ns or | 0.280413117 | s |
| Time elapsed was | 285786219 | ns or | 0.285786219 | s |
| Time elapsed was | 280018772 | ns or | 0.280018772 | s |
| Time elapsed was | 387279078 | ns or | 0.387279078 | s |
| Time elapsed was | 323816453 | ns or | 0.323816453 | s |
| Time elapsed was | 350336692 | ns or | 0.350336692 | s |
| Time elapsed was | 327693833 | ns or | 0.327693833 | s |
| Time elapsed was | 324254417 | ns or | 0.324254417 | s |
| Time elapsed was | 327702671 | ns or | 0.327702671 | s |
| Time elapsed was | 363039116 | ns or | 0.363039116 | s |
| Average mean time | 325034036.8 | ns or | 0.325034037 | s |

Write a short analysis of the results

Above table shows the mean running time of simple program and CUDA version of linear regression. The mean running time of linear program run in CPU is 74.7 ns and the CUDA program which runs in GPU is 325034036.8 ns. It is clear that the program run in CPU is faster than that of GPU. when we run the program in GPU we need to copy the data from CPU To GUP which takes more time. While doing it in the CPU version it requires minimum time as the program itself is very small. To conclude, if the numbers of data to be proceeding are lower than it will take more time even after running 1000 threads.

# 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
```