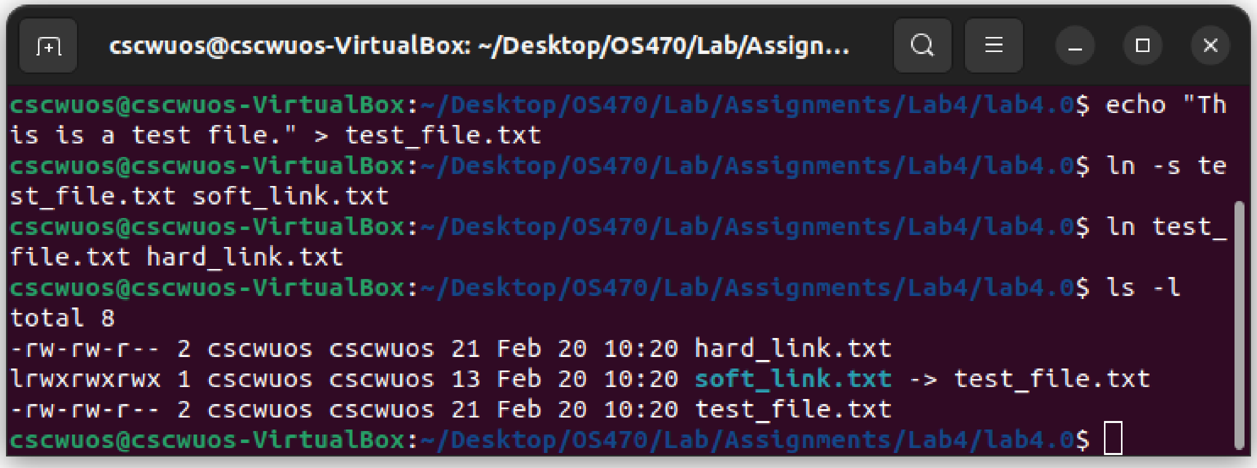
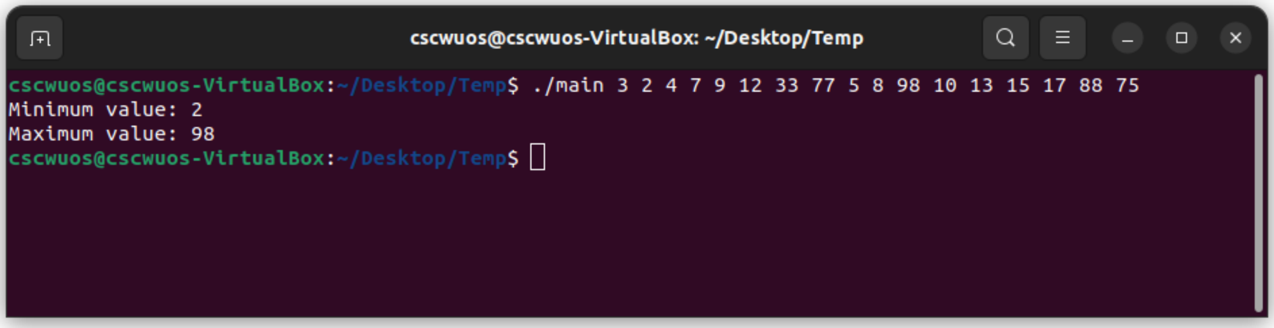
Questions:
1. Create hard, and soft links using the ln and ln -s commands.



```
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.0$ echo "Th
is is a test file." > test_file.txt
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.0$ ln -s te
st_file.txt soft_link.txt
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.0$ ln test_
file.txt hard_link.txt
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.0$ ls -l
total 8
-rw-rw-r-- 2 cscwuos cscwuos 21 Feb 20 10:20 hard_link.txt
lrwxrwxrwx 1 cscwuos cscwuos 13 Feb 20 10:20 soft_link.txt -> test_file.txt
-rw-rw-r-- 2 cscwuos cscwuos 21 Feb 20 10:20 test_file.txt
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.0$ 
```

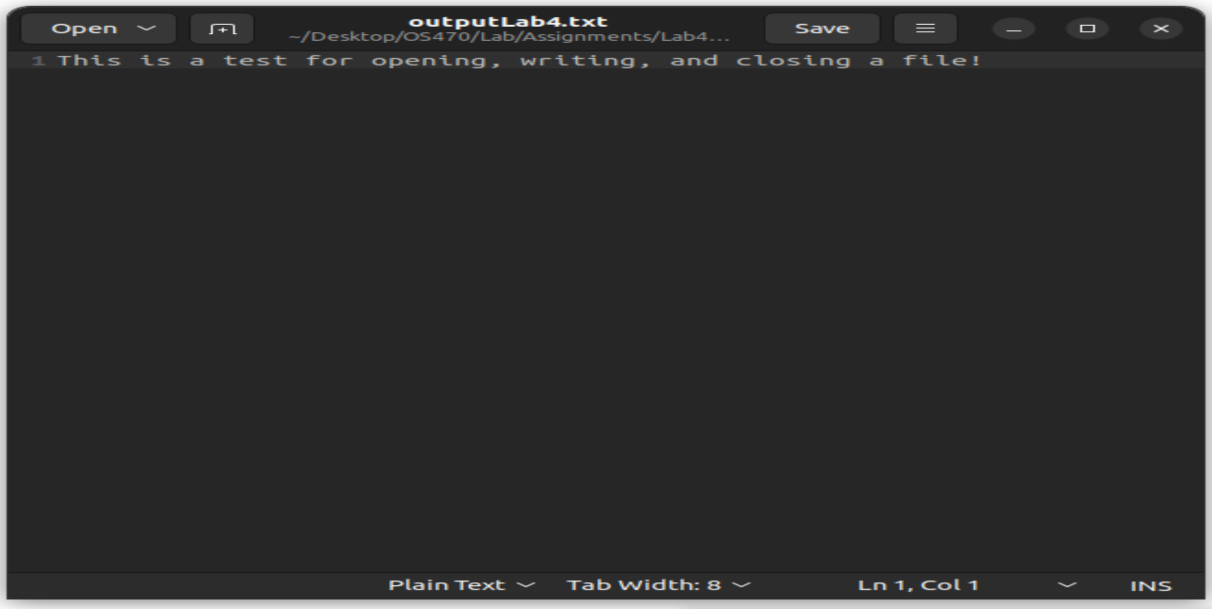
2. Create a multi-threaded program that computes different statistical values for a set of numbers. This application will start two independent worker threads when given a series of numbers on the command line. One thread will compute the greatest value, and the next will add the minimum value. Assume your program is given a list of integers. (The array of numbers must be provided as a parameter to the threads, and the thread must return the calculated value to the main thread.)



```
cscwuos@cscwuos-VirtualBox:~/Desktop/Temp$ ./main 3 2 4 7 9 12 33 77 5 8 98 10 13 15 17 88 75
Minimum value: 2
Maximum value: 98
cscwuos@cscwuos-VirtualBox:~/Desktop/Temp$ 
```

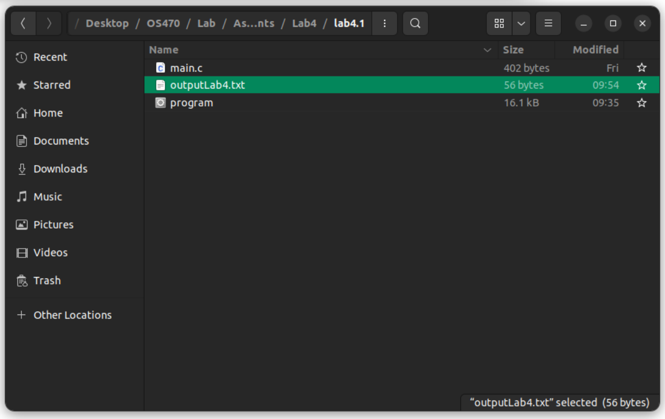
The program creates two threads, one for computing the MIN value and one for adding the MAX value of the array. The threads are passed a pointer to a struct obj which contains a pointer to the collection of numbers, the length of the array, and a field for storing the results. The min_worker and the max_worker iterate over the array finding and storing the minimum and maximum values into the result field of the thread_data and exiting. After creating the threads, the main thread waits for them to complete using pthread_join() and prints the results.

3. Write a C program that opens the file "outputLab4.txt" for writing and appends the phrase, "This is a test for opening, writing, and closing a file!"



```
This is a test for opening, writing, and closing a file!
```

```
#include <stdio.h>

int main() {
FILE *fp;
char phrase[] = "This is a test for opening, writing, and closing a file!";

fp = fopen("outputLab4.txt", "a"); // Open the file for appending

if (fp == NULL) {
printf("Error opening file\n");
return 1;
}

fprintf(fp, "%s", phrase); // Write the phrase to the file

fclose(fp); // Close the file

return 0;
}
```

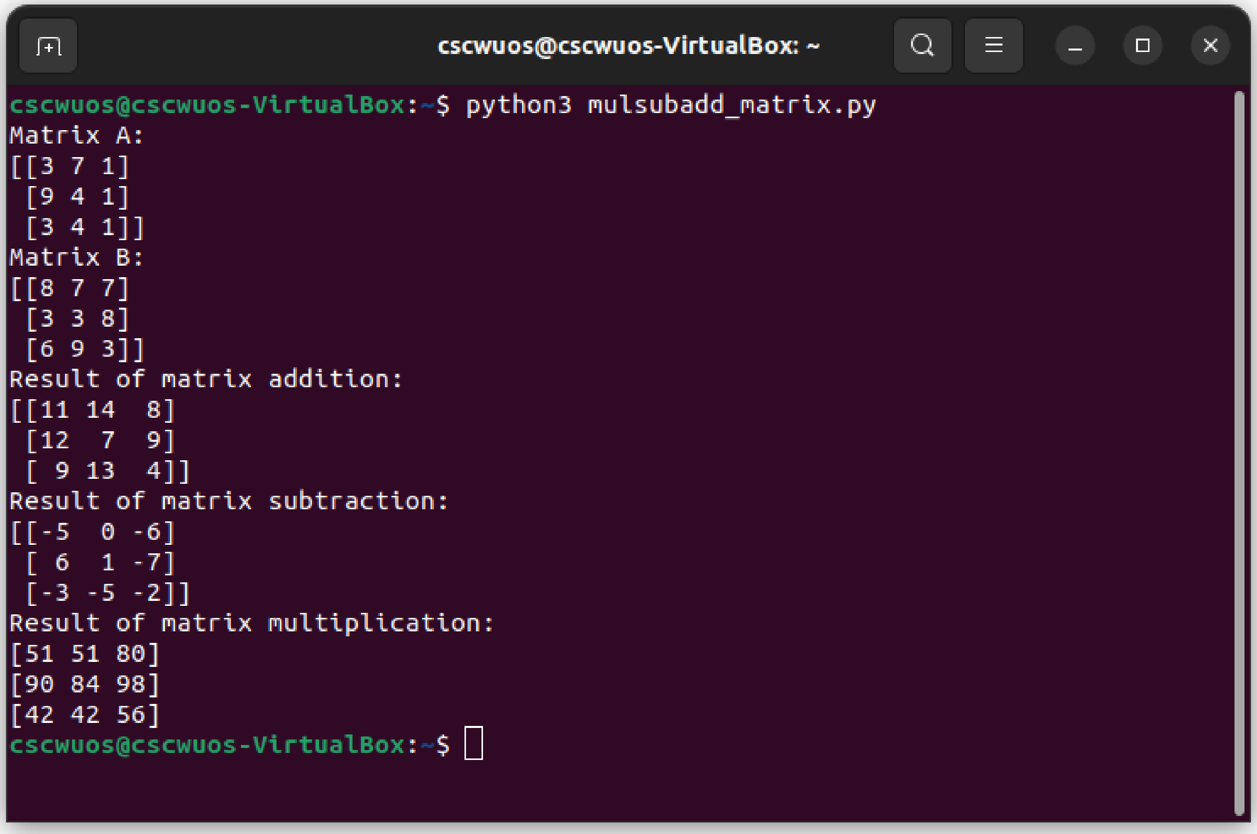


Name	Size	Modified
main.c	402 bytes	Fri
outputLab4.txt	56 bytes	09:54
program	16.1 kB	09:35

"outputLab4.txt" selected (56 bytes)

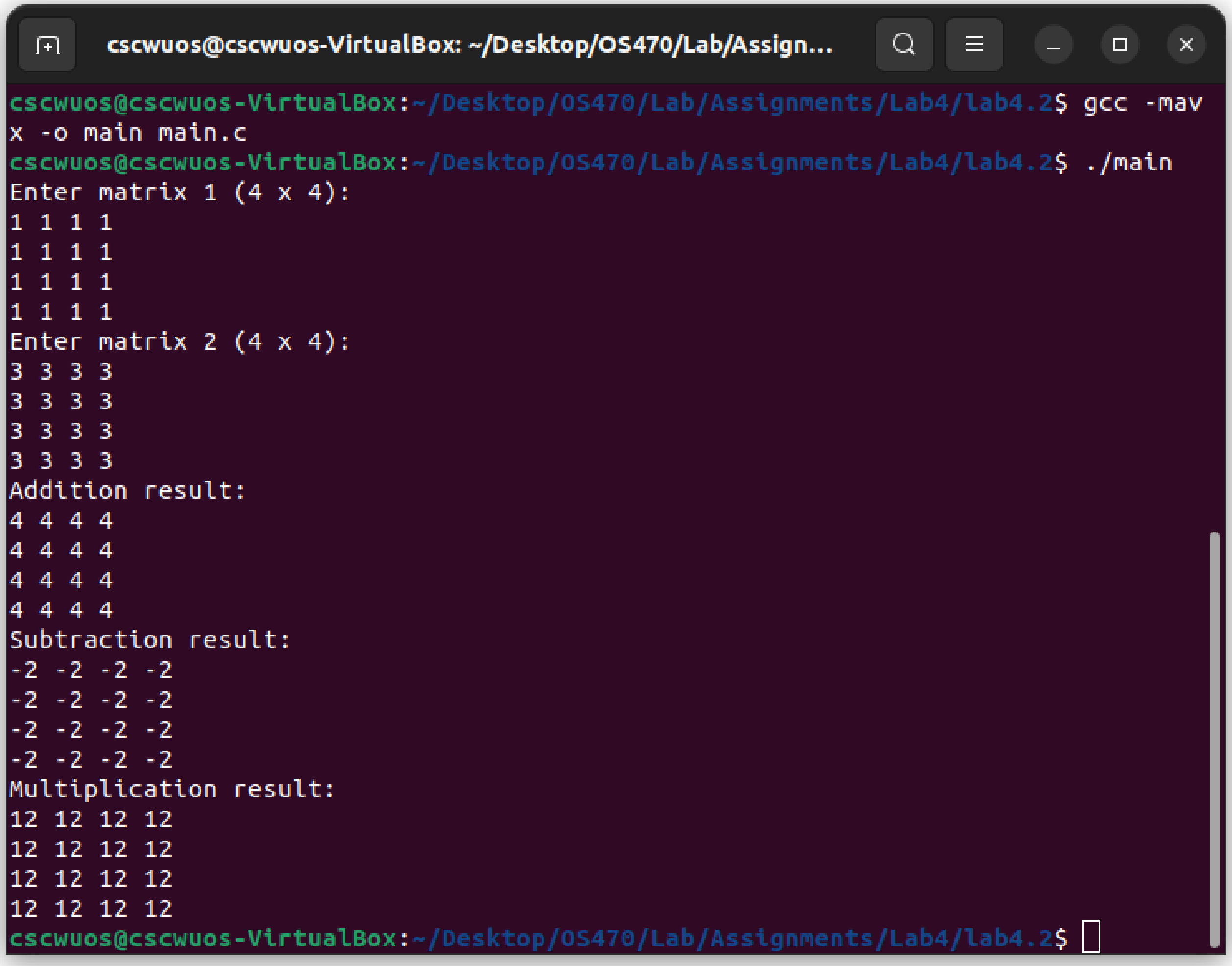
The program written in C opens an empty file name outputLab4.txt for appending, if no file is made an error exception is thrown. If the file is created then the phrase is appended and the file is saved as a text file. That is outputLab4.txt.

4. Write a program for matrix addition, subtraction, and multiplication using multi-threading.



```
cscwuos@cscwuos-VirtualBox:~$ python3 mulsubadd_matrix.py
Matrix A:
[[3 7 1]
 [9 4 1]
 [3 4 1]]
Matrix B:
[[8 7 7]
 [3 3 8]
 [6 9 3]]
Result of matrix addition:
[[11 14  8]
 [12  7  9]
 [ 9 13  4]]
Result of matrix subtraction:
[[-5  0 -6]
 [ 6  1 -7]
 [-3 -5 -2]]
Result of matrix multiplication:
[51 51 80]
[90 84 98]
[42 42 56]
cscwuos@cscwuos-VirtualBox:~$
```

Here I used python3 and the NumPy library to implement this program. NumPy contains functions to add, subtract, and multiply matrices quickly. In this implementation, we have insisted on whole numbers and used the functions dot, add, and subtract to solve our matrices. These powerful functions built into the NumPy libraries can be handy, especially when dealing with large matrix operations.



```
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.2$ gcc -mav
x -o main main.c
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.2$ ./main
Enter matrix 1 (4 x 4):
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
Enter matrix 2 (4 x 4):
3 3 3 3
3 3 3 3
3 3 3 3
3 3 3 3
Addition result:
4 4 4 4
4 4 4 4
4 4 4 4
4 4 4 4
Subtraction result:
-2 -2 -2 -2
-2 -2 -2 -2
-2 -2 -2 -2
-2 -2 -2 -2
Multiplication result:
12 12 12 12
12 12 12 12
12 12 12 12
12 12 12 12
cscwuos@cscwuos-VirtualBox:~/Desktop/OS470/Lab/Assignments/Lab4/lab4.2$ 
```

The C program performs matrix addition, subtraction, and multiplication. First, the matrices are initialized, and three threads are created to perform the matrix operations. Each thread is passed a pointer to a struct containing the operation to perform. After the threads complete their tasks, the resulting solutions are printed out. The program then frees the dynamically allocated memory and returns a 0 to indicate a successful program completion.