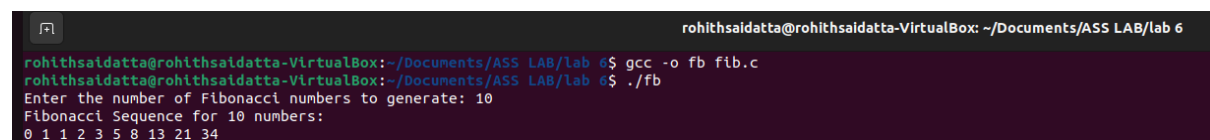


1. Write a multithreaded program that generates the Fibonacci series. The program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program is to generate. The program then will create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that is shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution the parent will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, this will require having the parent thread wait for the child thread to finish.

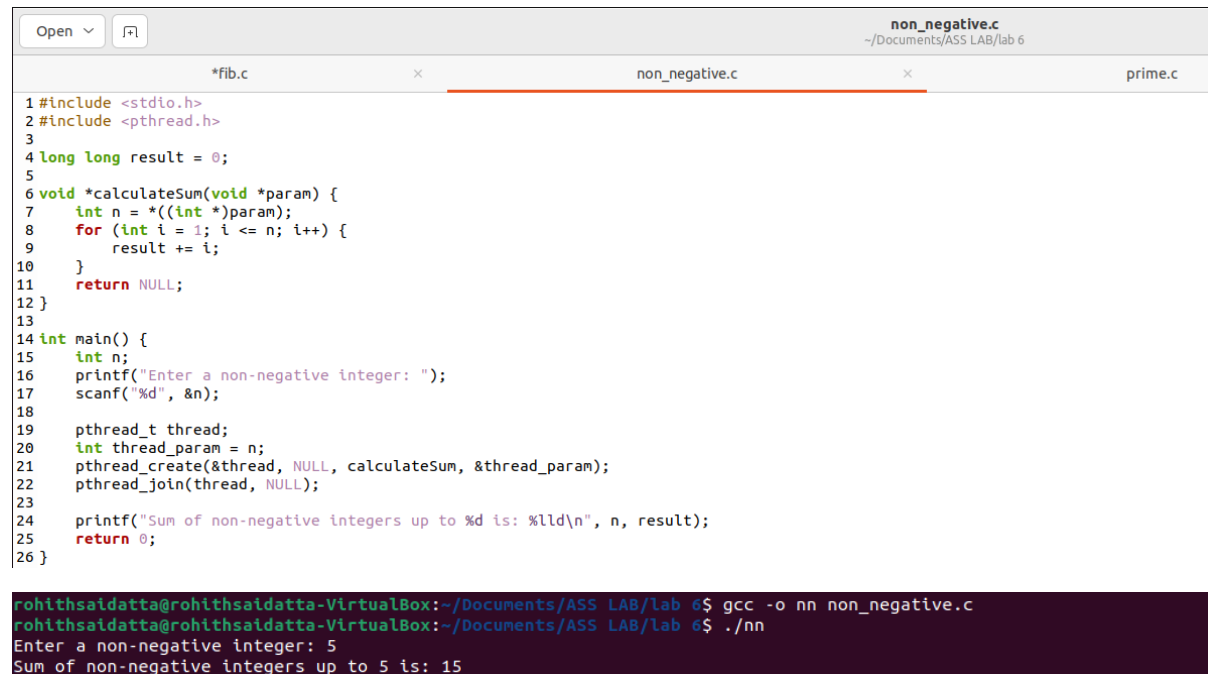


```
1 #include <stdio.h>
2 #include <pthread.h>
3 #define MAX_FIBONACCI 100
4 long long fibonacci[MAX_FIBONACCI];
5 void *generateFibonacci(void *param) {
6     int n = *((int *)param);
7     fibonacci[0] = 0;
8     fibonacci[1] = 1;
9     for (int i = 2; i < n; i++)
10         fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
11     return NULL;
12 }
13
14 int main() {
15     int n;
16     printf("Enter the number of Fibonacci numbers to generate: ");
17     scanf("%d", &n);
18
19     if (n <= 0 || n > MAX_FIBONACCI) {
20         printf("Invalid input. Please enter a positive integer <= %d.\n", MAX_FIBONACCI);
21         return 1;
22     }
23
24     pthread_t thread;
25     int thread_param = n;
26     pthread_create(&thread, NULL, generateFibonacci, &thread_param);
27     pthread_join(thread, NULL);
28     printf("Fibonacci Sequence for %d numbers:\n", n);
29     for (int i = 0; i < n; i++)
30         printf("%lld ", fibonacci[i]);
31     printf("\n");
32     return 0;
33 }
```



```
rohithsaidatta@rohithsaidatta-VirtualBox: ~/Documents/ASS LAB/lab 6
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ gcc -o fb fib.c
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ ./fb
Enter the number of Fibonacci numbers to generate: 10
Fibonacci Sequence for 10 numbers:
0 1 1 2 3 5 8 13 21 34
```

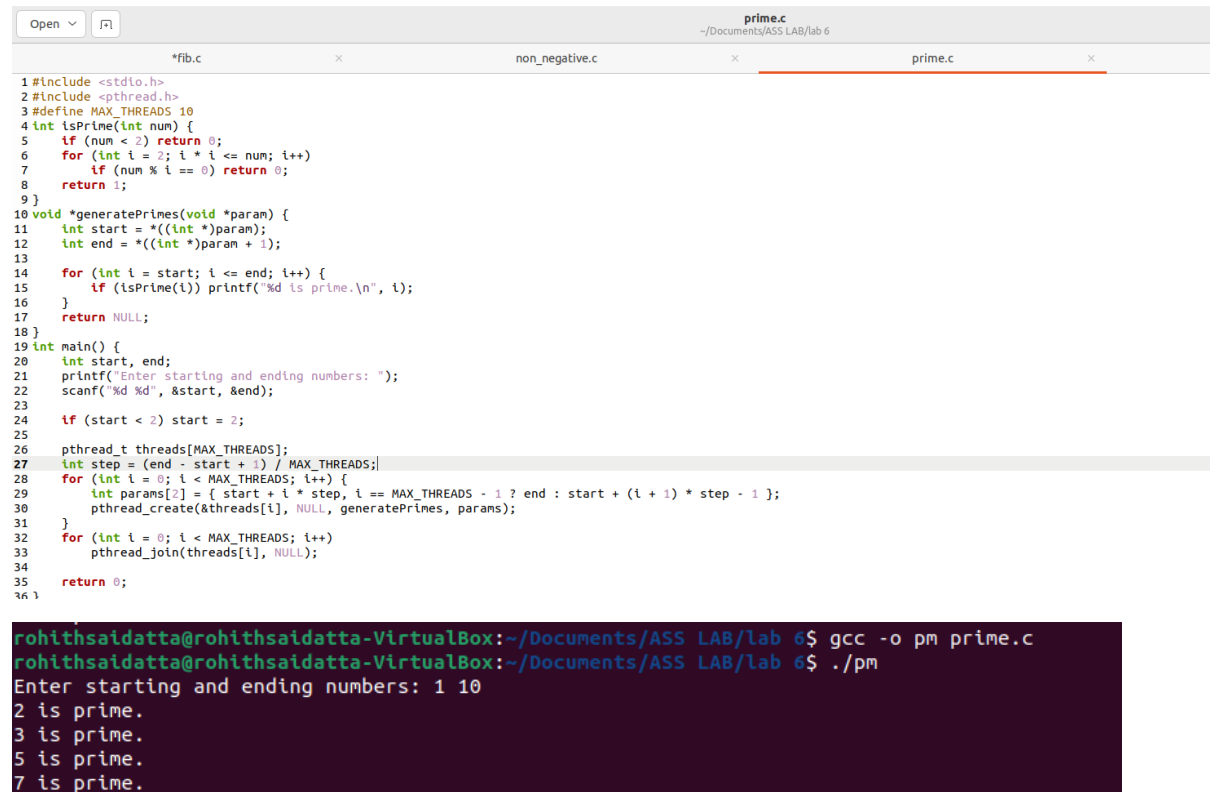
2. Write a multithreaded program that calculates the summation of non-negative integers in a separate thread and passes the result to the main thread.



```
1 #include <stdio.h>
2 #include <pthread.h>
3
4 long long result = 0;
5
6 void *calculateSum(void *param) {
7     int n = *((int *)param);
8     for (int i = 1; i <= n; i++) {
9         result += i;
10    }
11    return NULL;
12 }
13
14 int main() {
15     int n;
16     printf("Enter a non-negative integer: ");
17     scanf("%d", &n);
18
19     pthread_t thread;
20     int thread_param = n;
21     pthread_create(&thread, NULL, calculateSum, &thread_param);
22     pthread_join(thread, NULL);
23
24     printf("Sum of non-negative integers up to %d is: %lld\n", n, result);
25     return 0;
26 }
```

```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab «$ gcc -o nn non_negative.c
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab «$ ./nn
Enter a non-negative integer: 5
Sum of non-negative integers up to 5 is: 15
```

3. Write a multithreaded program for generating prime numbers from a given starting number to the given ending number.



```
1 #include <stdio.h>
2 #include <pthread.h>
3 #define MAX_THREADS 10
4 int isPrime(int num) {
5     if (num < 2) return 0;
6     for (int i = 2; i * i <= num; i++)
7         if (num % i == 0) return 0;
8     return 1;
9 }
10 void *generatePrimes(void *param) {
11     int start = *((int *)param);
12     int end = *((int *)param + 1);
13
14     for (int i = start; i <= end; i++) {
15         if (isPrime(i)) printf("%d is prime.\n", i);
16     }
17     return NULL;
18 }
19 int main() {
20     int start, end;
21     printf("Enter starting and ending numbers: ");
22     scanf("%d %d", &start, &end);
23
24     if (start < 2) start = 2;
25
26     pthread_t threads[MAX_THREADS];
27     int step = (end - start + 1) / MAX_THREADS;
28     for (int i = 0; i < MAX_THREADS; i++) {
29         int params[2] = { start + i * step, i == MAX_THREADS - 1 ? end : start + (i + 1) * step - 1 };
30         pthread_create(&threads[i], NULL, generatePrimes, params);
31     }
32     for (int i = 0; i < MAX_THREADS; i++)
33         pthread_join(threads[i], NULL);
34
35     return 0;
36 }
```

```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ gcc -o pm prime.c
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ ./pm
Enter starting and ending numbers: 1 10
2 is prime.
3 is prime.
5 is prime.
7 is prime.
```

4. Write a multithreaded program that performs the sum of even numbers and odd numbers in an input array. Create a separate thread to perform the sum of even numbers and odd numbers. The parent thread has to wait until both the threads are done.

```

*fib.c      non_negative.c      prime.c      *odd_even_sum.c
1 #include <stdio.h>
2 #include <pthread.h>
3 #define MAX_ARRAY_SIZE 100
4 #define MAX_THREADS 2
5 int array[MAX_ARRAY_SIZE];
6 long long evenSum = 0;
7 long long oddSum = 0;
8 int arraySize;
9 void *calculateEvenSum(void *param) {
10     for (int i = 0; i < arraySize; i++) {
11         if (array[i] % 2 == 0) {
12             evenSum += array[i];
13         }
14     }
15     pthread_exit(NULL);
16 }
17 void *calculateOddSum(void *param) {
18     for (int i = 0; i < arraySize; i++) {
19         if (array[i] % 2 != 0) {
20             oddSum += array[i];
21         }
22     }
23     pthread_exit(NULL);
24 }
25 int main() {
26     printf("Enter the size of the array: ");
27     scanf("%d", &arraySize);
28
29     printf("Enter %d integers:\n", arraySize);
30     for (int i = 0; i < arraySize; i++) {
31         scanf("%d", &array[i]);
32     }
33     pthread_t threads[MAX_THREADS];
34     pthread_create(&threads[0], NULL, calculateEvenSum, NULL);
35     pthread_create(&threads[1], NULL, calculateOddSum, NULL);
36
37     for (int i = 0; i < MAX_THREADS; i++) {
38         pthread_join(threads[i], NULL);
39     }
40     printf("Sum of even numbers: %lld\n", evenSum);
41     printf("Sum of odd numbers: %lld\n", oddSum);
42     return 0;
43 }

```

```

rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ gcc -o oe odd_even_sum.c
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/ASS LAB/lab 6$ ./oe
Enter the size of the array: 5
Enter 5 integers:
1 2 3 4 5
Sum of even numbers: 6
Sum of odd numbers: 9

```