190905514 **MOHAMMAD TOFIK** **BATCH : C3**
ROLLNO : 62 **SECTION : C** **SEM : 5th**

-: OS-LAB-6 :-

LAB EXCERCISE :

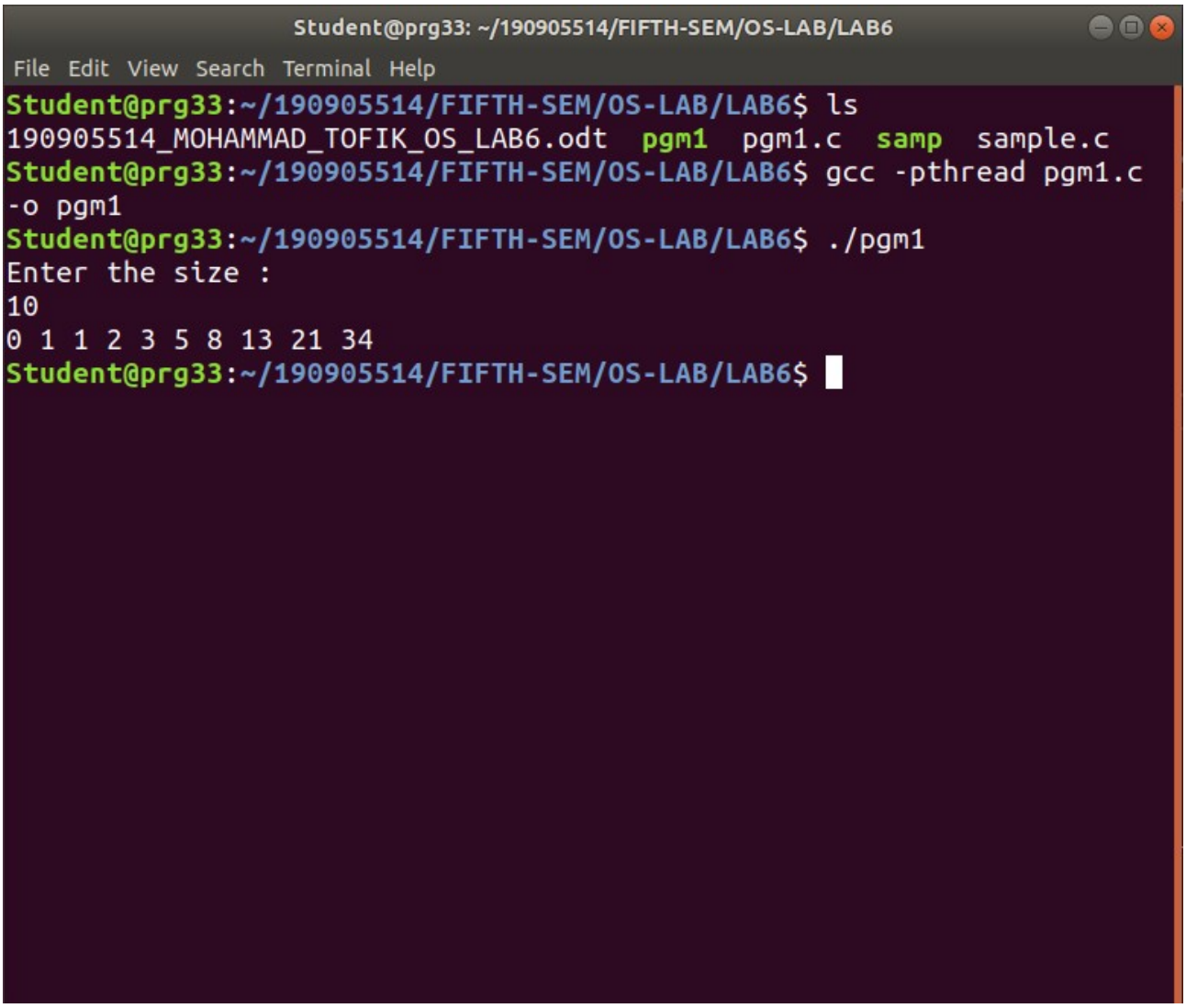
1. Write a multithreaded program that generates the Fibonacci series. The program should work as follows: The user will enter on the command line the number of Fibonacci numbers that the program is to generate. The program then will create a separate thread that will generate the Fibonacci numbers, placing 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.

pgm1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *generate_fibonacci(void *param)
{
int *arr = (int *)param;
int n = arr[0];
arr[1] = 0;
arr[2] = 1;

for (int i = 3; i <= n; i++)
{
arr[i] = arr[i - 1] + arr[i - 2];
}
return NULL;
}

int main(int argc, char const *argv[])
{
int n;
printf("Enter no of Fibonacci numbers : \n");
scanf("%d", &n);
```

```c
int *arr = (int *)malloc((n + 1) * sizeof(int));
arr[0] = n;

pthread_t thread;
pthread_create(&thread, 0, &generate_fibonacci, (void *)arr);
pthread_join(thread, 0);

for (int i = 1; i <= n; i++)
printf("%d ", arr[i]);
printf("\n");

return 0;
}
```

OUTPUT



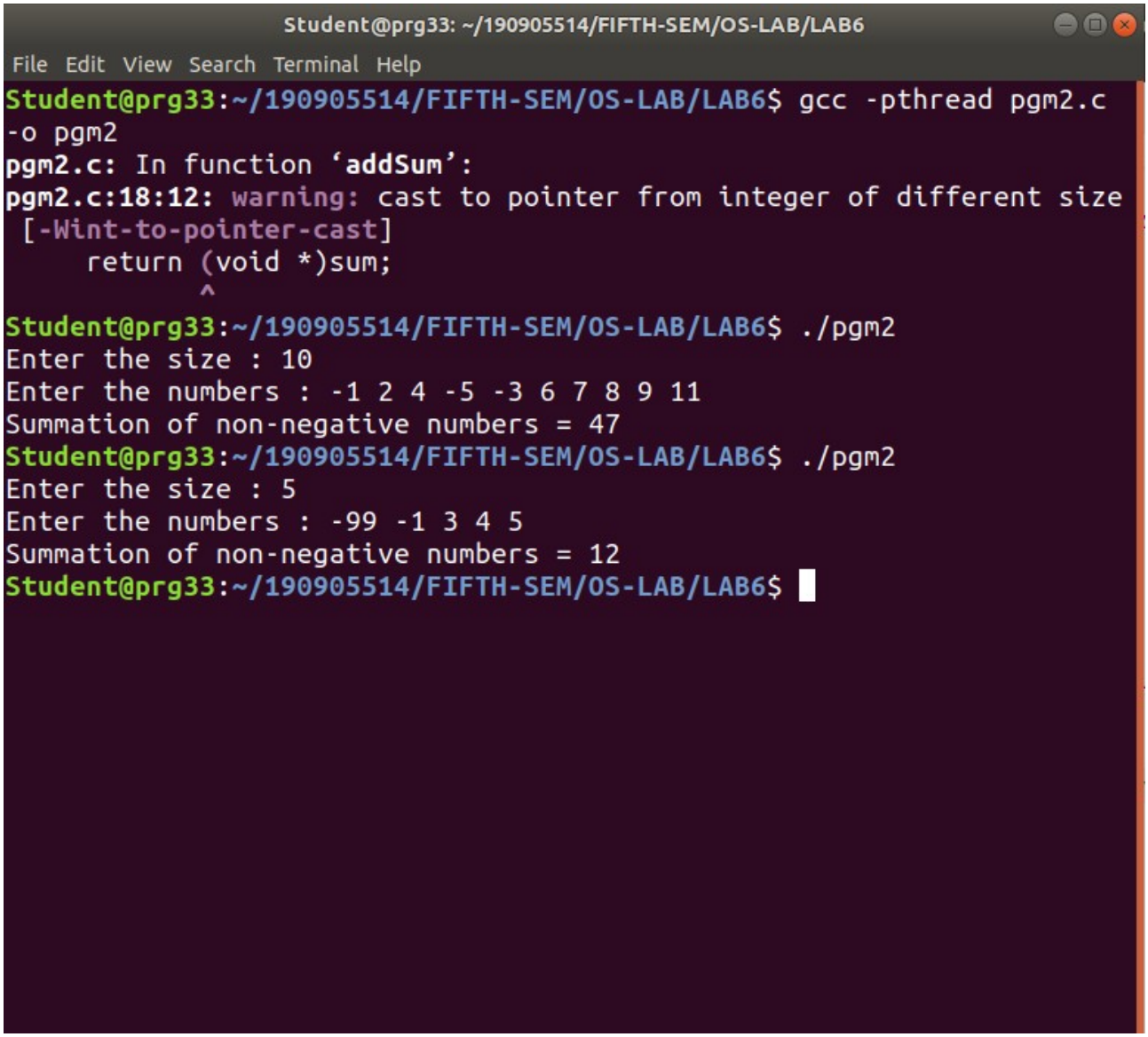
2.Write a multithreaded program that calculates the summation of non-negative integers in a separate thread and passes the result to the main thread.

pgm2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *addSum(void *parameter)
{
int *array = (int *)parameter;
int sum = 0;
int n = array[0];

for (int i = 1; i <= n; i++)
{
if (array[i] > 0)
sum += array[i];
}

return (void *)sum;
}

int main(int argc, char const *argv[])
{
int n;
printf("Enter the size : ");
scanf("%d", &n);

int *array = (int *)malloc((n + 1) * sizeof(int));

array[0] = n;

printf("Enter the numbers : ");

for (int i = 1; i <= n; i++)
{
scanf("%d", &array[i]);
}
```

```c
int answer = 0;
pthread_t thread;
pthread_create(&thread, 0, &addSum, (void *)array);
pthread_join(thread, (void **)&answer);

printf("Summation of non-negative numbers = %d\n", answer);

return 0;
}
```

OUTPUT



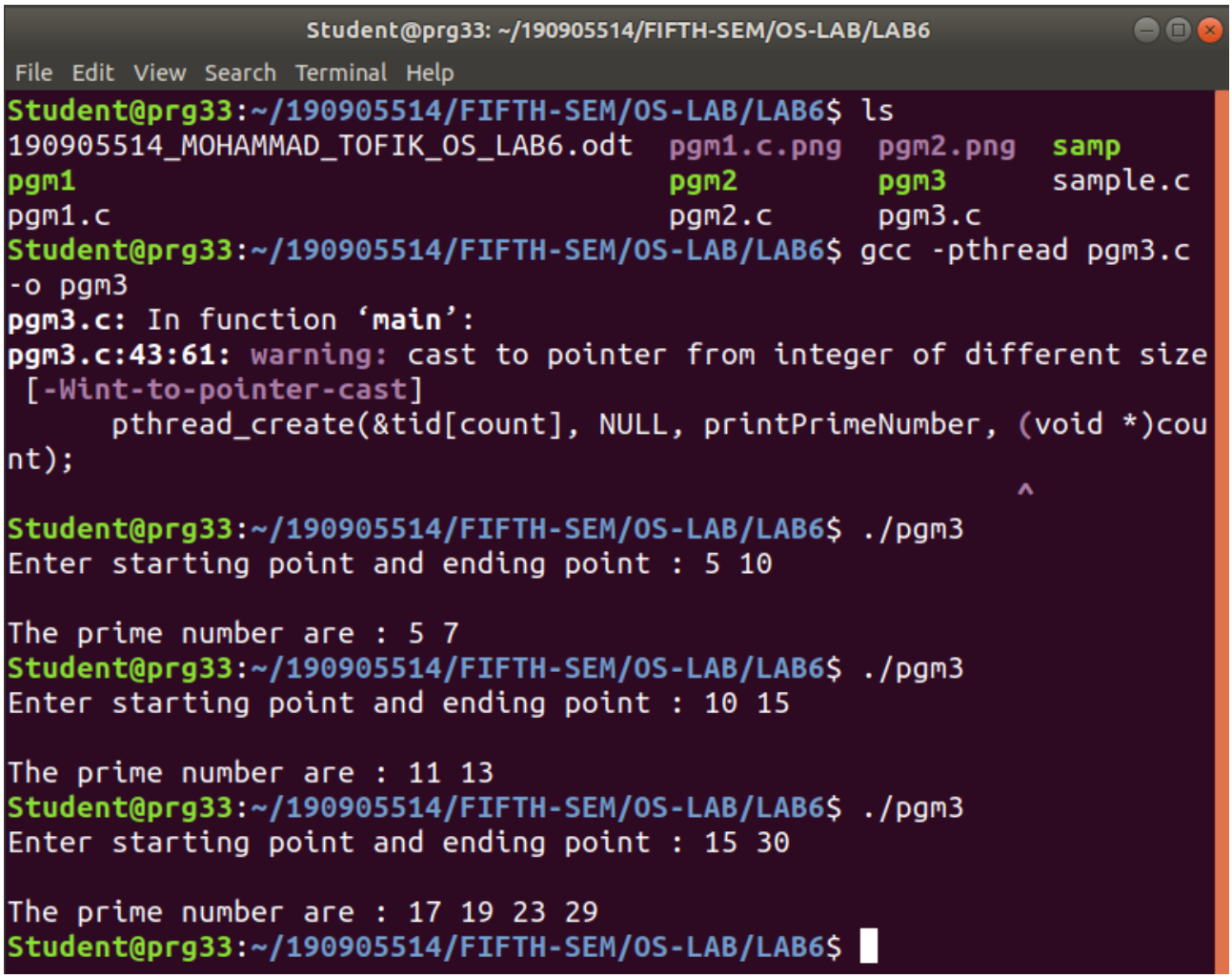
3. Write a multithreaded program for generating prime numbers from a given starting number to the given ending number.

pgm3.c

```
#include <stdio.h>
#include <pthread.h>
#define N 30
#define MAX_THREADS 4
int primeArray[N] = {0};
void *printPrimeNumber(void *ptr)
{
int j, flag;
int i = (int)(long long int)ptr;
while (i < N)
{
flag = 0;
for (j = 2; j <= i / 2; j++)
{
if (i % j == 0)
{
flag = 1;
break;
}
}
if (flag == 0 && (i > 1))
{
primeArray[i] = 1;
}
i += MAX_THREADS;
}
}
int main()
{
pthread_t tid[MAX_THREADS] = {0};
int count = 0;
printf("Enter starting point and ending point : ");
int startingPoint, endingPoint;
scanf("%d %d", &startingPoint, &endingPoint);
for (count = 0; count < MAX_THREADS; count++)
```

```
{
pthread_create(&tid[count], NULL, printPrimeNumber, (void *)count);
}
printf("\n");
for (count = 0; count < MAX_THREADS; count++)
{
pthread_join(tid[count], NULL);
}
int c = 0;
printf("The prime number are : ");
for (count = startingPoint; count < endingPoint; count++)
if (primeArray[count] == 1)
printf("%d ", count);
printf("\n");
return 0;
}
```

OUTPUT



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.

pgm4.c

```
#include <stdio.h>
#include <pthread.h>
int a[100];
void *thread_code_even(void *param)
{
int n = *(int *)param;
int sum = 0;
for (int i = 0; i < n; ++i)
{
if (a[i] % 2 == 0)
{
sum += a[i];
}
}
return (void *)sum;
}
void *thread_code_odd(void *param)
{
int n = *(int *)param;
int sum = 0;
for (int i = 0; i < n; ++i)
{
if (a[i] % 2 != 0)
{
sum += a[i];
}
}
return (void *)sum;
}
int main()
{
pthread_t thread1, thread2;
int n;
printf("Enter the size:\n");
```

```
scanf("%d", &n);
printf("Enter the elements:\n");
for (int i = 0; i < n; ++i)
{
scanf("%d", &a[i]);
}
int sumeven, sumodd;
pthread_create(&thread1, 0, &thread_code_even, &n);
pthread_join(thread1, (void **)&sumeven);
pthread_create(&thread2, 0, &thread_code_odd, &n);
pthread_join(thread2, (void **)&sumodd);
printf("Even sum: %d\n", sumeven);
printf("Odd sum: %d\n", sumodd);
printf("%d is the sum of array\n", sumeven + sumodd);
}
```

OUTPUT