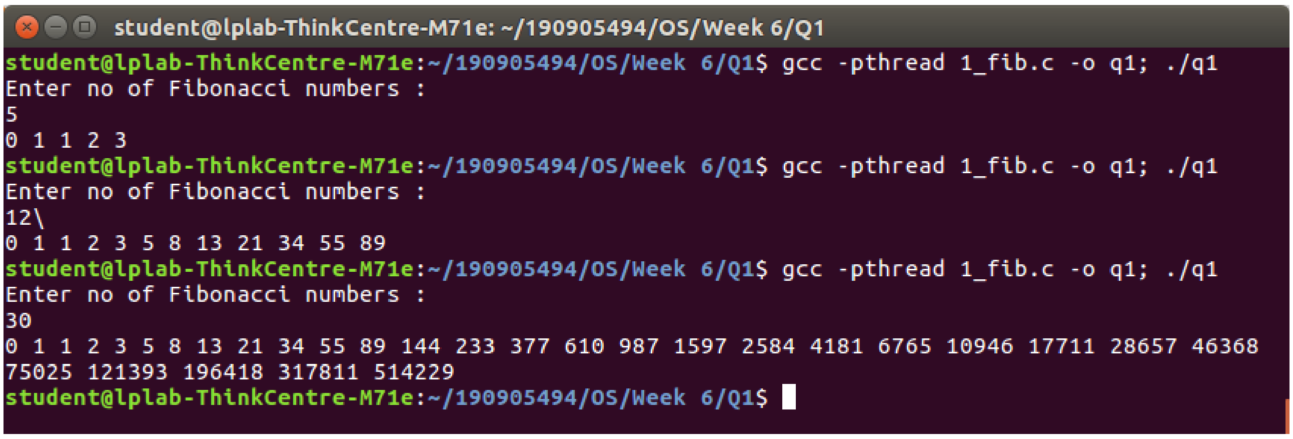
OS Lab #6

Question 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.
*/

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

void* generate_fibonacci(void* param)
{
  // initializing array and values
  int* arr = (int*)param;
  int n = arr[0];
  arr[1] = 0;
  arr[2] = 1;

  // creating fibonacci series in array
  for(int i = 3;i <= n;i++)
  {
    arr[i] = arr[i-1] + arr[i-2];
  }
  return NULL;
}

int main(int argc, char const *argv[])
{
  // getting the number of fibbonaci number we are to print
  int n;
  printf("Enter no of Fibonacci numbers : \n");
  scanf("%d",&n);

  // dynamically initializing our array
  int* arr = (int*)malloc((n+1)*sizeof(int));
  arr[0] = n;

  // creating thread
  pthread_t thread;

  // running routine and passing array
  pthread_create(&thread,0,&generate_fibonacci,(void*)arr);
  // joining child threads to the main thread
  pthread_join(thread,0);


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

  return 0;
}
```

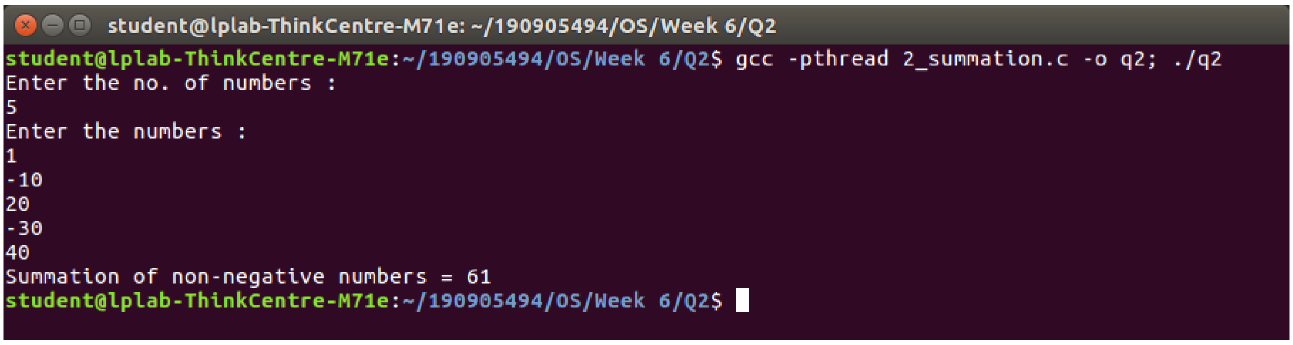


student@lplab-ThinkCentre-M71e: ~/190905494/OS/Week 6/Q1

```
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q1$ gcc -pthread 1_fib.c -o q1; ./q1
Enter no of Fibonacci numbers :
5
0 1 1 2 3
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q1$ gcc -pthread 1_fib.c -o q1; ./q1
Enter no of Fibonacci numbers :
12\
0 1 1 2 3 5 8 13 21 34 55 89
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q1$ gcc -pthread 1_fib.c -o q1; ./q1
Enter no of Fibonacci numbers :
30
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
75025 121393 196418 317811 514229
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q1$
```

Question 2

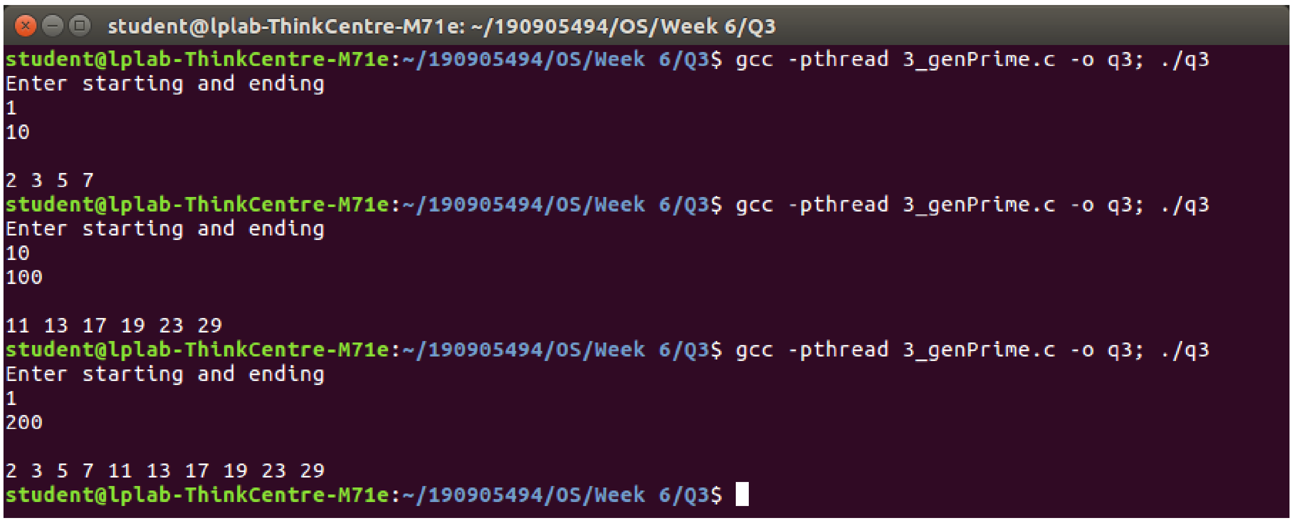
```
/*
Write a multithreaded program that calculates the summation
of non-negative integers in a separate thread and passes the
result to the main thread.
*/


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

void* summation(void *param)
{
        // initializing array
        int* arr = (int*)param;
        long sum = 0;
        int n = arr[0];

        // summing and returning
        for(int i = 1;i <= n;i++) {
                if(arr[i] > 0)
                        sum += arr[i];
        }

        return (void*)sum;
}

int main(int argc, char const *argv[])
{
        // initializing and getting the numbers
        int n;
        printf("Enter the no. of numbers : \n");
        scanf("%d",&n);

        // dynamically initializing our array
        int* arr = (int*)malloc((n+1)*sizeof(int));
        arr[0] = n;

        // entering values
        printf("Enter the numbers : \n");
        for(int i= 1;i <= n;i++)
        {
                scanf("%d",&arr[i]);
        }

        int answer = 0;

        // creating thread
        pthread_t thread;
        // running routine and passing array
        pthread_create(&thread,0,&summation,(void*)arr);

        // joining child threads to the main thread
        pthread_join(thread,(void**)&answer);

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

        return 0;
}
```



```
student@lplab-ThinkCentre-M71e: ~/190905494/OS/Week 6/Q2
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q2$ gcc -pthread 2_summation.c -o q2; ./q2
Enter the no. of numbers :
5
Enter the numbers :
1
-10
20
-30
40
Summation of non-negative numbers = 61
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q2$
```

Question 3

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

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

#define N 30
#define MAX_THREADS 4

int prime_arr[N]={0};

void *printprime(void *ptr)
{
   // initializing values
   int  j,flag;
   int i=(int)(long long int)ptr;

   // finding prime numbers
   while(i<N)
   {
      printf("Thread id[%d] checking [%d]\n",pthread_self(),i);
      flag=0;
      for(j=2;j<=i/2;j++)
      {
          if(i%j==0)
          {
              flag=1;
              break;
          }
      }

      if(flag==0 && (i>1))
      {
          prime_arr[i]=1;
      }
      i+=MAX_THREADS;
}
}

int main()
{
   // initializing values and threads
   pthread_t tid[MAX_THREADS]={0};
   long count=0;

   // enterning min and max values
   printf("Enter starting and ending\n");
   int st,en;
   scanf("%d %d",&st,&en);

   for(count=0;count<MAX_THREADS;count++)
   {
      printf("\r\n CREATING THREADS %d",count);
      // running routine and passing array
      pthread_create(&tid[count],NULL,printprime,(void*)count);
   }
   printf("\n");

   for(count=0;count<MAX_THREADS;count++)
   {
      // joining child threads to the main thread
      pthread_join(tid[count],NULL);
   }

   int c=0;

   // printing solution
   for(count=st;count<en;count++)
      if(prime_arr[count]==1)
          printf("%ld ",count);
   printf("\n");

   return 0;
}
```

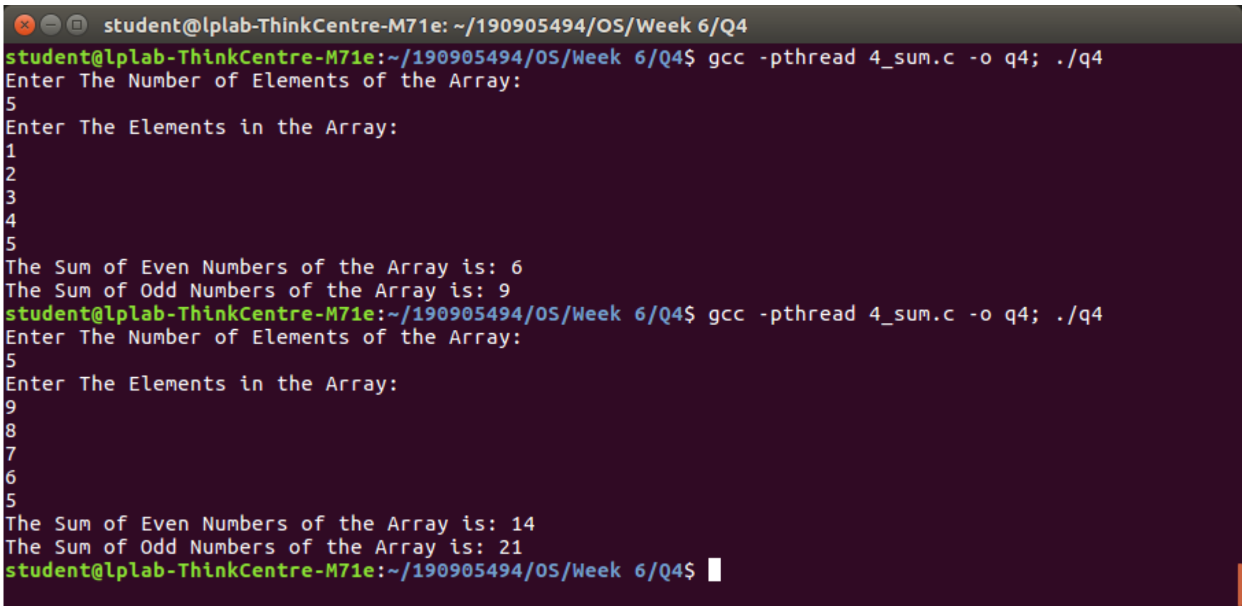


```
student@lplab-ThinkCentre-M71e: ~/190905494/OS/Week 6/Q3
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q3$ gcc -pthread 3_genPrime.c -o q3; ./q3
Enter starting and ending
1
10

2 3 5 7
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q3$ gcc -pthread 3_genPrime.c -o q3; ./q3
Enter starting and ending
10
100

11 13 17 19 23 29
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q3$ gcc -pthread 3_genPrime.c -o q3; ./q3
Enter starting and ending
1
200

2 3 5 7 11 13 17 19 23 29
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q3$
```

Question 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.
*/

#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>


void *even(void *brr)
{
   // initializing array and values
   int *arr = (int *)brr;
   int size = arr[0];
   long sum = 0;

   // finding even sum
   for (int i = 1; i <= size; i++)
      if (arr[i] % 2 == 0)
         sum += arr[i];
   return (void *)sum;
}
void *odd(void *brr)
{
   // initializing array and values
   int *arr = (int *)brr;
   int size = arr[0];
   long sum = 0;

   // finding odd sum
   for (int i = 1; i <= size; i++)
```

```
        if (arr[i] % 2 != 0)
            sum += arr[i];
    return (void *)sum;
}


int main()
{
    // initializing values
    int n, evenSum, oddSum;

    // getting array information
    printf("Enter The Number of Elements of the Array: \n");
    scanf("%d", &n);
    int arr[n + 1];
    arr[0] = n;
    printf("Enter The Elements in the Array:\n");
    for (int i = 1; i <= n; i++)
        scanf("%d", &arr[i]);

    // creating thread
    pthread_t t1, t2;

    // running routine and passing array
    pthread_create(&t1, 0, &even, (void *)arr);
    pthread_create(&t2, 0, &odd, (void *)arr);

    // joining child threads to the main thread
    pthread_join(t1, (void *)&evenSum);
    pthread_join(t2, (void *)&oddSum);

    // showing outputs
    printf("The Sum of Even Numbers of the Array is: %d\n", (int)evenSum);
    printf("The Sum of Odd Numbers of the Array is: %d\n", (int)oddSum);
}
```



```
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q4$ gcc -pthread 4_sum.c -o q4; ./q4
Enter The Number of Elements of the Array:
5
Enter The Elements in the Array:
1
2
3
4
5
The Sum of Even Numbers of the Array is: 6
The Sum of Odd Numbers of the Array is: 9
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q4$ gcc -pthread 4_sum.c -o q4; ./q4
Enter The Number of Elements of the Array:
5
Enter The Elements in the Array:
9
8
7
6
5
The Sum of Even Numbers of the Array is: 14
The Sum of Odd Numbers of the Array is: 21
student@lplab-ThinkCentre-M71e:~/190905494/OS/Week 6/Q4$
```