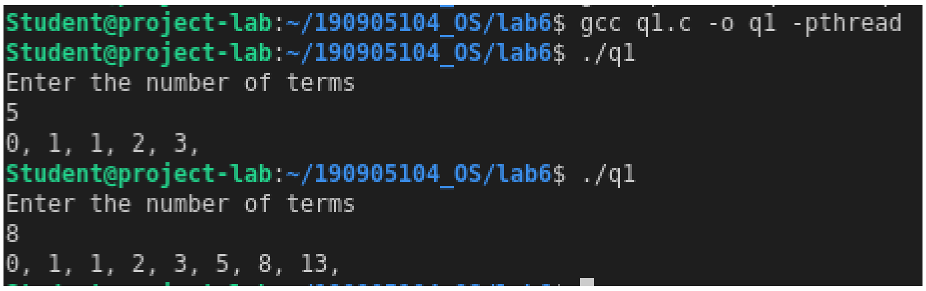
190905104
Parth Shukla
Lab 6

Programs on Threads

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.

```
// multithreaded fibonacci
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void* gen(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* argv[]){
    int n;
    printf("Enter the number of terms\n");
    scanf("%d", &n);

    int arr[100];
    for(int i = 0; i <= n; i++){
        arr[i] = 0;
    }
    arr[0] = n;

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

    for(int i = 1; i <= n; i++){
        printf("%d, ", arr[i]);
    }
    printf("\n");
    return 0;
```

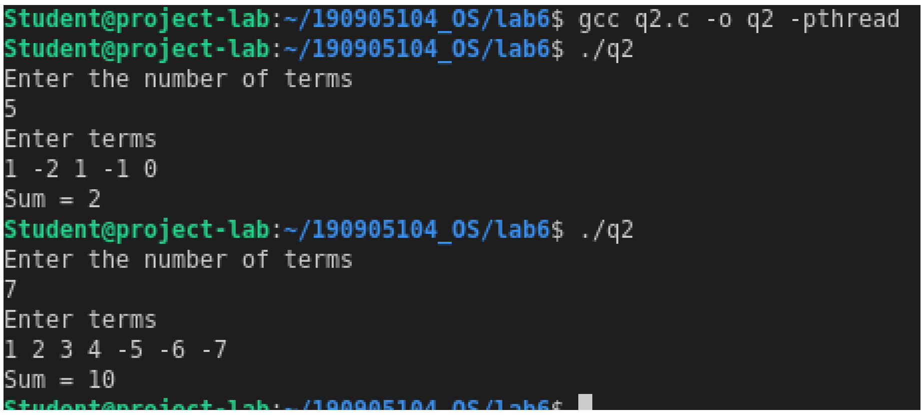
}



```
Student@project-lab:~/190905104_OS/lab6$ gcc q1.c -o q1 -pthread
Student@project-lab:~/190905104_OS/lab6$ ./q1
Enter the number of terms
5
0, 1, 1, 2, 3,
Student@project-lab:~/190905104_OS/lab6$ ./q1
Enter the number of terms
8
0, 1, 1, 2, 3, 5, 8, 13,
```

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.

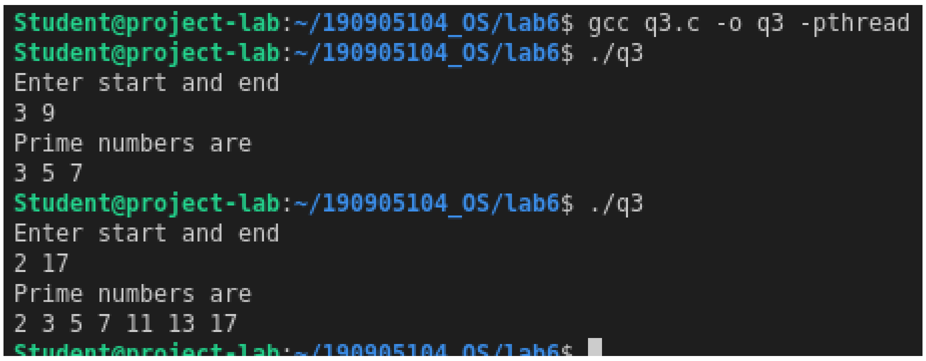
```
// summation non negative
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void* calc(void* param){
    int* arr = (int *)param;
    int n = arr[0];
    int sum = 0;
    for(int i = 1; i <= n; i++){
        if(arr[i] >= 0)
            sum += arr[i];
    }
    arr[n+1] = sum;
    return NULL;
}

int main(int argc, char* argv[]){
    int n;
    printf("Enter the number of terms\n");
    scanf("%d", &n);

    int arr[100];
    for(int i = 0; i <= n+1; i++){
        arr[i] = 0;
    }
    arr[0] = n;
    printf("Enter terms\n");
    for(int i = 1; i <= n; i++){
        scanf("%d", &arr[i]);
    }

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

    printf("Sum = %d\n", arr[n+1]);
    return 0;
}
```



```
Student@project-lab:~/190905104_OS/lab6$ gcc q2.c -o q2 -pthread
Student@project-lab:~/190905104_OS/lab6$ ./q2
Enter the number of terms
5
Enter terms
1 -2 1 -1 0
Sum = 2
Student@project-lab:~/190905104_OS/lab6$ ./q2
Enter the number of terms
7
Enter terms
1 2 3 4 -5 -6 -7
Sum = 10
Student@project-lab:~/190905104_OS/lab6$
```

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

```
// prime numbers

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

int isprime(int n){
    for(int i = 2; i < n; i++){
        if(n % i == 0){
            // not prime
            return 0;
        }
    }
    return 1;
}

void* prime(void* param){
    int* arr = (int *)param;
    int count=0;
    int start = arr[0];
    int end = arr[1];
    for(int i = start; i <= end; i++){
        if(isprime(i) == 1){
            arr[3+count] = i;
            count++;
        }
    }
    arr[2] = count;
    return NULL;
}

int main(int argc, char * argv[]){

    int arr[100];
    for(int i = 0; i <= 100; i++){
        arr[i] = 0;
    }

    printf("Enter start and end\n");
    scanf("%d %d", &arr[0], &arr[1]);
    pthread_t thread;
    pthread_create(&thread, 0, &prime, (void *)arr);
    pthread_join(thread, 0);
    printf("Prime numbers are\n");
    int count = arr[2];
    for(int i = 0; i < count; i++){
        printf("%d ", arr[3+i]);
    }
    printf("\n");
    return 0;
}
```

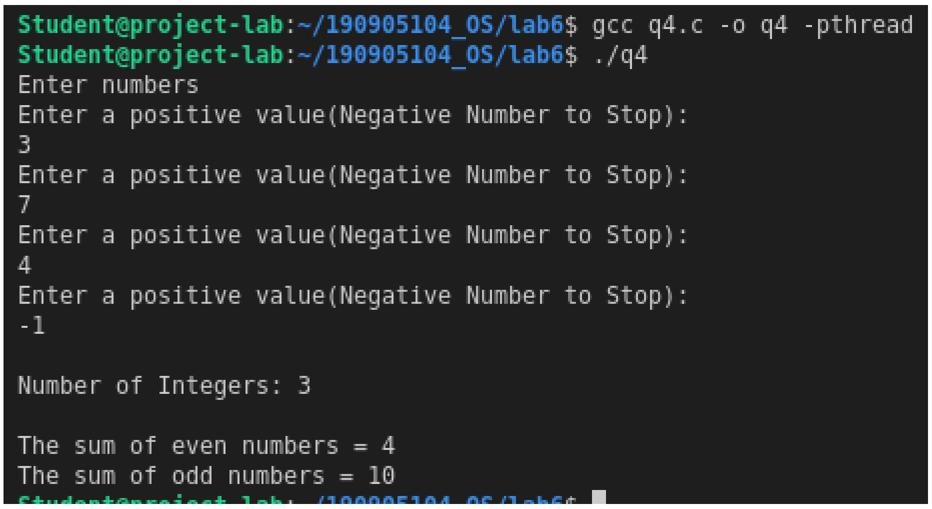


```
Student@project-lab:~/190905104_OS/lab6$ gcc q3.c -o q3 -pthread
Student@project-lab:~/190905104_OS/lab6$ ./q3
Enter start and end
3 9
Prime numbers are
3 5 7
Student@project-lab:~/190905104_OS/lab6$ ./q3
Enter start and end
2 17
Prime numbers are
2 3 5 7 11 13 17
Student@project-lab:~/190905104_OS/lab6$
```

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

volatile int running_threads = 0;
pthread_t thread[1];
int numOfElements;
int evensum, oddsum;

void *findsum(void *array_ptr){
    int i;

    int *arr = (int*)array_ptr;
    for(i = 0; i < numOfElements; i++){
        if(arr[i]%2==0)
            evensum += arr[i];
        else
            oddsum += arr[i];
    }

    running_threads -= 1;

    return NULL;
}
```

```
int getArrayInput(int n, int *arr_ptr){
    int input;
    int numberOfElements = 0;

          printf("Enter numbers\n");

    while(1){
        printf("Enter a positive value(Negative Number to Stop):\n");

        if(scanf("%d",&input) != 1){
            printf("Invalid\n");
            exit(1);
        }

          if(input >= 0){
                    if (numberOfElements == n){
                              n += 1;
                              arr_ptr = realloc(arr_ptr, n * sizeof(int));
                    }

          arr_ptr[numberOfElements++] = input;
        }

        else{
          printf("\nNumber of Integers: %d\n", numberOfElements);
          break;
        }
    }

    return numberOfElements;
}


void createThreads(int *arr_ptr){
    int s;
    s = pthread_create(&thread[2], NULL, findsum, (void *)arr_ptr);

    running_threads += 1;
}

int main(){
    int n = 1;
    int *arr_ptr = malloc(n * sizeof(int));
    numOfElements = getArrayInput(n, arr_ptr);

    createThreads(arr_ptr);

    while(running_threads>0){
        sleep(1);
    }

    printf("\nThe sum of even numbers = %d\nThe sum of odd numbers = %d\n", evensum,
oddsum);
    return(0);
}
```



```
Student@project-lab:~/190905104_OS/lab6$ gcc q4.c -o q4 -pthread
Student@project-lab:~/190905104_OS/lab6$ ./q4
Enter numbers
Enter a positive value(Negative Number to Stop):
3
Enter a positive value(Negative Number to Stop):
7
Enter a positive value(Negative Number to Stop):
4
Enter a positive value(Negative Number to Stop):
-1

Number of Integers: 3

The sum of even numbers = 4
The sum of odd numbers = 10
Student@project-lab:~/190905104_OS/lab6$
```