

LAB EXERCISES:

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 will then create a separate thread that will generate 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 child thread to finish.

Program:

```
#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);

    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:

```

@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB6
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ gcc -pthread fibo.c -o fibo
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ ./fibo
Enter no of Fibonacci numbers :
10
0 1 1 2 3 5 8 13 21 34
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ █

```

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

Program:

```

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

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

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

    return (void*)sum;
}

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

    printf("Enter the no. of numbers : \n");
    scanf("%d",&n);

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

```

```

arr[0] = n;

printf("Enter the numbers : \n");

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

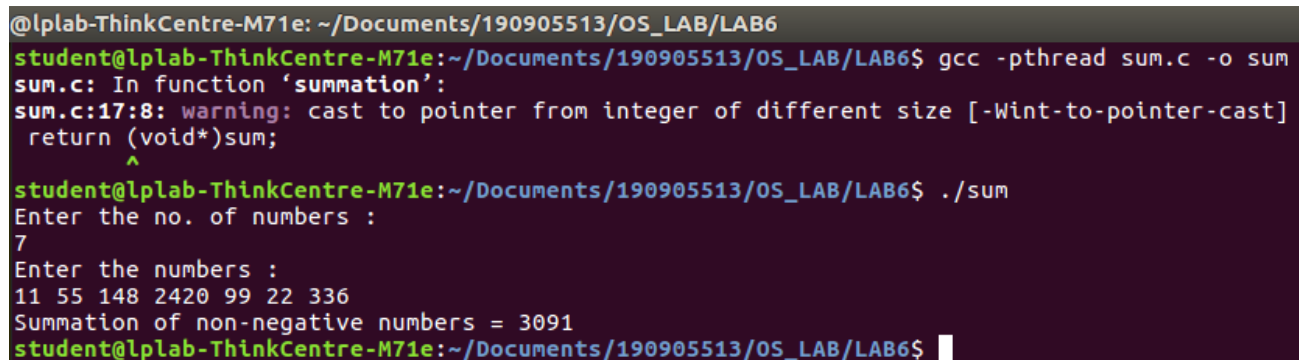
int answer = 0;
pthread_t thread;
pthread_create(&thread,0,&summation,(void*)arr);
pthread_join(thread,(void**)&answer);

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

return 0;
}

```

Output:



```

@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB6
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ gcc -pthread sum.c -o sum
sum.c: In function 'summation':
sum.c:17:8: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    return (void*)sum;
           ^
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ ./sum
Enter the no. of numbers :
7
Enter the numbers :
11 55 148 2420 99 22 336
Summation of non-negative numbers = 3091
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$

```

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

Program:

```

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

#define N 9999
#define MAX_THREADS 4

int prime_arr[N]={0};

void *printprime(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))
    {
        prime_arr[i]=1;
    }
    i+=MAX_THREADS;
}
}

int main()
{
    pthread_t tid[MAX_THREADS]={0};
    int count=0;
    printf("Enter starting and ending\n");
    int st,en;
    scanf("%d %d",&st,&en);

    for(count=0;count<MAX_THREADS;count++)
    {
        pthread_create(&tid[count],NULL,printprime,(void*)count);
    }
    printf("\n");
    for(count=0;count<MAX_THREADS;count++)
    {
        pthread_join(tid[count],NULL);
    }

    int c=0;

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

    return 0;
}

```

Output:

```
@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB6
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ gcc -pthread genpr.c -o genpr
genpr.c: In function 'main':
genpr.c:36:5: warning: braces around scalar initializer
    pthread_t tid[MAX_THREADS]={0};
    ^
genpr.c:36:5: note: (near initialization for 'tid[0]')
genpr.c:44:52: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    pthread_create(&tid[count],NULL,printprime,(void*)count);
                                                         ^
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ ./genpr
Enter starting and ending
100 200

101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ ./genpr
Enter starting and ending
1 10

2 3 5 7
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/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.

Program:

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <string.h>
void *even(void *brr)
{
    int *arr = (int *)brr;
    int size = arr[0];
    int sum = 0;
    for (int i = 1; i <= size; i++)
        if (arr[i] % 2 == 0)
            sum += arr[i];
    return (void *)sum;
}
void *odd(void *brr)
{
    int *arr = (int *)brr;
    int size = arr[0];
    int sum = 0;
    for (int i = 1; i <= size; i++)
        if (arr[i] % 2 != 0)
            sum += arr[i];
    return (void *)sum;
}
int main()
{
```

```

int n, evenSum, oddSum;
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]);
pthread_t t1, t2;
pthread_create(&t1, 0, &even, (void *)arr);
pthread_create(&t2, 0, &odd, (void *)arr);
pthread_join(t1, (void *)&evenSum);
pthread_join(t2, (void *)&oddSum);
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);
}

```

Output:

```

student@lplab-ThinkCentre-M71e: ~/Documents/190905513/OS_LAB/LAB6
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ gcc -pthread sumoe.c -o sumoe
sumoe.c: In function 'even':
sumoe.c:22:12: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    return (void *)sum;
           ^
sumoe.c: In function 'odd':
sumoe.c:32:12: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
    return (void *)sum;
           ^
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$ ./sumoe
Enter The Number of Elements of the Array:
10
Enter The Elements in the Array:
1 2 3 4 5 6 7 8 9 10
The Sum of Even Numbers of the Array is: 30
The Sum of Odd Numbers of the Array is: 25
student@lplab-ThinkCentre-M71e:~/Documents/190905513/OS_LAB/LAB6$

```