COMPSCI 3SH3 Winter, 2021
Student name: Khizar Siddiqui
Student ID: 400109902
Date: 05-March-2021

Lab 3 Report

Textbook Question 4.22

```
#include  <pthread.h>
#include  <stdio.h>
#include  <stdlib.h>
#define  NUMBER_OF_THREADS 3

void *averageFinder(void *params);      /*  thread  that  calculates  average  */
void *minimumFinder(void *params);      /*  thread  that  calculates  minimum  */
void *maximumFinder(void *params);      /*  thread  that  calculates  maximum  */
int argCount, avg, min, max;

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

    argCount = argc - 1;
    int arguments[argCount];
    for (int i = 0; i < argCount; i++) {
        arguments[i] = atoi(argv[i + 1]);
    }
    pthread_t workers[NUMBER_OF_THREADS];
    pthread_create(&workers[0], 0, averageFinder, (void *) &arguments);
    pthread_create(&workers[1], 0, minimumFinder, (void *) &arguments);
    pthread_create(&workers[2], 0, maximumFinder, (void *) &arguments);
    for (int i = 0; i < NUMBER_OF_THREADS; i++) {
        pthread_join(workers[i], NULL);
    }
    printf("The average value is %d \n", avg);
    printf("The minimum value is %d \n", min);
    printf("The maximum value is %d \n", max);
}

void *averageFinder(void *params) {
    int *l = (int *) params;
    int sum = 0;
    for (int i = 0; i < argCount; i++) {
        sum += l[i];
    }
    avg = sum/argCount;
    pthread_exit(0);
}

void *minimumFinder(void *params) {
    int *l = (int *) params;
    min = l[0];
    for (int i = 1; i < argCount; i++) {
        if (l[i] < min) {
            min = l[i];
        }
    }
    pthread_exit(0);
}

void *maximumFinder(void *params) {
    int *l = (int *) params;
    max = l[0];
    for (int i = 1; i < argCount; i++) {
        if (l[i] > max) {
            max = l[i];
        }
    }
    pthread_exit(0);
}
```

My code creates 3 threads as well as 3 functions and 3 global variables for each of these threads. Using a loop over the argument count, we increment our list of numbers (entered as input) and pass this parameter to each function. The functions (average, maximum, minimum) calculate their respective values and modify the global variables accordingly. Once all the threads are complete and exit, we join these threads back in and print the information out.

Textbook Question 4.23

```
#include  <pthread.h>
#include  <stdio.h>
#include  <stdlib.h>
#define  NUMBER_OF_THREADS 1
```

```
void *primeFinder(void *params);      /* thread that calculates prime */
int primeCheck(int num);             /* helper function that checks prime*/
int input;

int main(int argc, char * argv[]) {
    input = atoi(argv[1]);
    pthread_t workers[NUMBER_OF_THREADS];
    pthread_create(&workers[0], 0, primeFinder, (void *) &input);
    pthread_join(workers[0], NULL);
}

void *primeFinder(void *params) {
    int *val = (int *) params;
    int n = *val;
    for (int i = 2; i <= n; i++) {
        if (primeCheck(i)) {
            printf("%d ", i);
        }
    }
    printf("\n");
    pthread_exit(0);
}

int primeCheck(int num) {
    int res = 1;
    for (int i = 2; i < num; i++) {
        if (num % i == 0) {
            res = 0;
        }
    }
    return res;
}
```

My code creates a single thread and a function (along with a personal helper function) to find all prime numbers before the entered argument. This entered argument is passed as a parameter to the function where it loops over numbers from 2 (smallest prime number) to the entered input and checks whether they are prime and prints them out if they are. The helper function loops from 2 to entered argument and checks whether those numbers can divide entered argument. If so, it is considered prime.