Academic Year	**2023**		
Semester	☐ Fall	☒ Winter	☐ Summer
Course Code - Name	CSCI 3310 – System Programming		
Instructor	Dr. Razi Iqbal		
Assessment	Assignment 1		
Deadline	Feb. 12, 2023		

Question 1

Threads in modern operating systems are used as a mean of improving the efficiency of the programs and avoiding lags in performance. Most of the modern operating systems provide threading using Thread Libraries. Some programming languages as Java also provide libraries for managing threads that can be incorporated in to programs to enhance the performance of the programs.

In this assessment, you are required to demonstrate the use of threading and provide an evidence of performance boost of the program using the built-in threading libraries. The assessment is two-folds. Below are the requirements of the assessment:

Fold 1:

You are required to write a program in C that takes (1,000,000,000) as an input (you are free to take this input from command line arguments or from the user). Once the input is taken, there should be 3 threads in your program performing the following operations:

1. Child Thread 1:
    a. Displays its ID
    b. Find incremental sum of all the numbers until and including 1,000,000,000
    c. Exit itself once completed the operation

2. Child Thread 2:
    a. Displays its ID
    b. Find incremental sum of all the even numbers until and including 1,000,000,000
    c. Exit itself once completed the operation

3. Parent / Main Thread:
    a. Displays its ID
    b. Creates Child Threads 1 and 2
    c. Merges Child Thread 1 and 2
    d. Displays the sum of all the numbers returned by Child Thread 1
    e. Displays the sum of all the numbers returned by Child Thread 2
    f. Calculates the following:
        i. sum of all the numbers returned by Child Thread 1 / sum of all the numbers returned by Child Thread 2
    g. Displays the result of above calculation

Requirement 1:

Analyze the time taken by the program to complete, e.g., review the usage of cores using Activity Monitor program in Ubuntu and find a way to monitor the time taken by the program to complete its execution. Furthermore, make sure, the calculations made by the program are correct and accurate. Do NOT forget to take a screenshot of the Activity Monitor.

Fold 2:
You are required to write a program in C that takes (1,000,000,000) as an input (you are free to take this input from command line arguments or from the user). Once the input is taken, the program should perform the following operations:

1. Find incremental sum of all the numbers until and including 1,000,000,000
2. Find incremental sum of all the even numbers until and including 1,000,000,000
3. Display the sum of all the numbers until and including 1,000,000,000
4. Displays the sum of all the even numbers until and including 1,000,000,000
5. Calculates the following:
    - sum of all the numbers in Step 1 / sum of all the numbers in Step 2
6. Displays the result of above calculation

Requirement 2:
Analyze the time taken by the program to complete, e.g., review the usage of cores using Activity Monitor program in Ubuntu and find a way to monitor the time taken by the program to complete its execution. Do NOT forget to take a screenshot of the Activity Monitor.

Requirement 3:
Compare the results obtained by the analysis of Requirement 1 and Requirement 2. Make sure to write down the reasons of your analysis in plain English.

Instructions:

In order to obtain maximum marks in this assignment, please ensure the followings:

- Submit this assignment by writing your solution in this document under the Solution heading below. Do not use a separate document. Only PDF submissions are accepted.
- Copy and paste the code from your editor in this document for both the folds.
- Make sure to take a screenshot of the output of both the folds.
- **Make sure you take the screenshot of the Activity Monitor for Requirement 1 and 2.**
- **For Requirement 3, make sure to provide a concrete reason in plain English.**
- This assignment has a weightage of **10%** marks of the course.
- This is **NOT** a group assignment so **students having similar assignments will get a 0.**
- The assignment deadline is **midnight Feb. 12, 2023**. Submissions after the deadline will not be accepted.

Solution

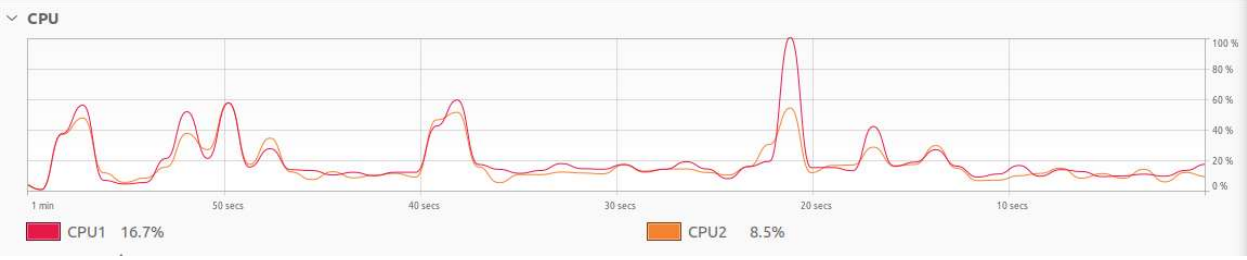
Fold 1

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

int num;

void* EvenSum(){
    unsigned long evensum = 0;
    printf("Second child thread ID: %lu \n", pthread_self());

    for (int i = 0; i<=num; i++){
        if (i%2 == 0){
            evensum += i;
        }
    }

    return (void *)evensum;
}

void* Sum(){
    unsigned long sum;
    printf("First child thread ID: %lu \n", pthread_self());
    for (int i = 0; i<=num; i++){
        sum += i;
    }

    return (void *)sum;
}

int main(int args, char* argv[]){
    num = atoi(argv[1]);
    printf("Main thread ID: %lu \n", pthread_self());

    pthread_t tid[2];

    pthread_create(&tid[0], NULL, EvenSum, NULL);
    pthread_create(&tid[1], NULL, Sum, NULL);

    void *sum = NULL;
    void *evensum = NULL;
    pthread_join(tid[0], &evensum);
    pthread_join(tid[1], &sum);

    double divValue = ((unsigned long)sum/1.0) / ((unsigned long)evensum/1.0);

    printf("Incremental sum of all the numbers: %lu\n",(unsigned long)sum);
    printf("Incremental sum of all the Even numbers: %lu \n",(unsigned long)evensum);
    printf("Divided value of both sums: %.1lf \n", divValue);

    return 0;
}
```

Output:

```
accr@SystemsProgram:~/Documents/Systems/ass1$ ./fold1.o 1000000000
Main thread ID: 140149265758016
First child thread ID: 140149257360960
Second child thread ID: 140149265753664


Incremental sum of all the numbers: 500000000500000000
Incremental sum of all the Even numbers: 250000000500000000
Divided value of both sums: 2.0
```

Activity monitor



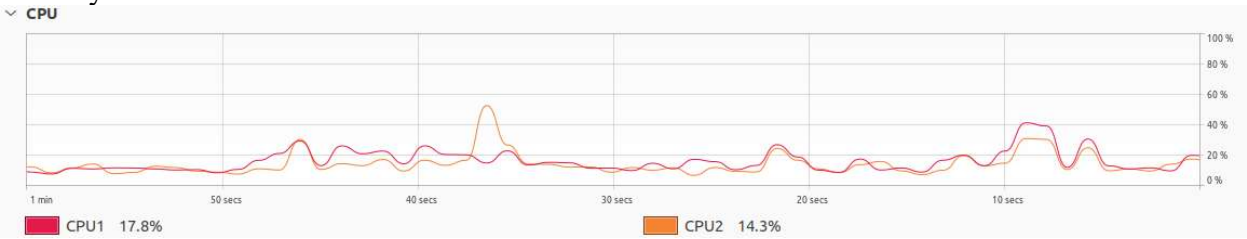
Fold 2

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

int main(int args,char *argv[]){
    int num = atoi(argv[1]);
    unsigned long sum = 0, evensum = 0;

    for (int i = 1; i <= num; i++){
        sum += i;
        if (i%2 == 0){
            evensum += i;
        }
    }

    double divValue = (sum/1.0)/(evensum/1.0);

    printf("Incremental sum of all the numbers: %lu\n",sum);
    printf("Incremental sum of all the Even numbers: %lu\n",evensum);
    printf("Divided value of both sums: %.1lf \n", divValue);

    return 0;
}
```

Output:

```
accr@SystemsProgram:~/Documents/Systems/ass1$ ./fold2.o 1000000000
Incremental sum of all the numbers: 500000000500000000
Incremental sum of all the Even numbers: 250000000500000000
Divided value of both sums: 2.0
```

Activity monitor



Requirement 3

Looking at the 2 activity monitors from both fold 1 and 2 we can see that fold 2 requires much less time to execute compare to fold 1. This is most likely due to fold 1 having to create 2 child thread to calculate the sum along with outputting everything on the main thread. Unlike fold 2 which easily computes everything in the single main function. This is proven by the large spikes in the activity from fold 1 compared to the streamline less demanding activity monitor from fold 1. In conclusion fold 2 requires less performance compared to fold 2.