Assignment 1: Starting with OpenMP

Exercise 1: Setting up the Environment

Ensure you have a C/C++ compiler that supports OpenMP installed on your system. Just open the text editor or IDE and start writing your code:

```
#include <stdio.h>
#include <omp.h>

int main() {
    #pragma omp parallel
    {   int thread_id = omp_get_thread_num();
        printf("Hello, World! This is thread %d\n", thread_id);
    }
    return 0;
}
```

Compile the code using:

```
gcc -o hello_openmp -fopenmp hello_openmp.c
```

Task 1

1. Provide a list of run-time routines that are used in OpenMP
2. Why aren't you seing the Hello World output thread sequence as 0, 1, 2, 3 etc. Why are they disordered?
3. What happens to the thread_id if you change its scope to before the pragma?
4. Convert the code to serial code.

Task 2

1. The following code adds two arrays of size 16 together and stores answer in result array.

```
#include <stdio.h>

int main() {

    int array1[16] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, …, 16};
    int array2[16] = {16, ..., 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

    int result[16];

    for (int i = 0; i < 16; i++) {
        result[i] = array1[i] + array2[i];
    }

    for (int i = 0; i < 16; i++) {
        printf("%d ", result[i]);
    }
    printf("\n");
    return 0;
}
```

2. Convert it into Parallel, such that only the addition part is parallelized.
3. The display loop at the end displays the result. Modify the code such that this is also parallel, but only thread of id 0 is able to display the entire loop. The others should not do anything. When making it parallel, make sure its the old threads and new threads are not created. What output do you see?

Task 3

1. Modify the code of Task 2 and do the job in half of the threads.

Task 4

1. The following code adds the contents of the array array1 and array2.

```
#include <stdio.h>

int main() {

    int array1[16] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, …, 16};
    int array2[16] = {16, ..., 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1};
    int result1 = 0, result2 = 0;

    for (int i = 0; i < 16; i++) {
        result1 += array1[i];
    }

    if (result1 > 10) {
        result2 = result1;
        for (int i = 0; i < 16; i++) {
          result2 += array2[i];
        }
    }

    printf("%d\n", result2);
    return 0;
}
```

2. Convert it into Parallel using 16 threads.
3. Try removing the reduction() clause and add **#pragma omp atomic** just beore the +=. What is the effect on result? Explain.

Task 5

1. Go to chatGPT and ask it to generate any C based code of 3-5 functions for testing gprof.
2. Profile the code using gprof

```
gcc gprof_test.c -p
```

3. Display the output of the gprof and its callgraph.
4. Use the gprof2dot utility https://github.com/jrfonseca/gprof2dot and convert the callgraph into a nice diagram.

Deliverable

Your deliverable should contain the following:

1. Your Name (Roll Number)
2. Your Source Code (with Comments)
3. The plot generated from your code (you can use excel), or a tool called gnuplot.