Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2022-23 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 1

Exam Seat No:

2019BTECS00055 – Aditya Rajendra Manapure

Title of practical: Introduction to OpenMP

Problem Statement 1:
To Implement a parallel code to print “Hello World” using OpenMP

Screenshot 1:

Assignment-1 > C++ Que1_hello.cpp > ...

```cpp
//Hello World using OpenMP
//Author : Aditya Manapure
//2019BTECS00055

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

#define THREAD_COUNT 8

int main(){

    double start = omp_get_wtime();     // start time
    omp_set_num_threads(THREAD_COUNT);  // fixing number of threads
    #pragma omp parallel
    {
        printf("thread No. %d  Hello World\n", omp_get_thread_num());
    }
    double end = omp_get_wtime();
    double total_time = end - start;
    printf("Time taken : %lf\n", total_time);
    return 0;

}
```

Screenshot 2:

```
C:\Users\adity\OneDrive\Documents\College\sem 7\HPC\HPC Lab\Assignment-1>g++ -fopenmp Que1_hello.cpp

C:\Users\adity\OneDrive\Documents\College\sem 7\HPC\HPC Lab\Assignment-1>a.exe
thread No. 5  Hello World
thread No. 1  Hello World
thread No. 0  Hello World
thread No. 6  Hello World
thread No. 3  Hello World
thread No. 4  Hello World
thread No. 2  Hello World
thread No. 7  Hello World
Time taken : 0.009000
```

Information 1:

When the master thread reaches this line, it forks additional threads to carry out the work enclosed in the block following the #pragma construct. The block is executed by all threads in parallel. The original thread will be denoted as master thread with thread-id 0.Once the compiler encounters the parallel regions code, the master thread (thread which has thread id 0) will fork into the specified number of threads. Entire code within the parallel region will be executed by all threads concurrently. Once the parallel region ended, all threads will get merged into the master thread.

Problem Statement 2:
To implement a parallel code to print squares of first 100 numbers followed by addition of all these squares.

Screenshot 1:

Assignment-1 > Que2_square.cpp > main(int, char * [])

```
//squares of first 100 numbers followed by addition of all these squares
//Author : 2019BTECS00055
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>

#define THREAD_COUNT 10

int main(int argc, char* argv[]){
    int i=1;
    int sum = 0;
    int square = 0;
    double start = omp_get_wtime();
    omp_set_num_threads(THREAD_COUNT);
    #pragma omp parallel
    {
        #pragma omp single
        printf("No. of Threads Allocated : %d \n", omp_get_num_threads());

        #pragma omp for schedule(static, 10)
        for(i=1; i<=100; i++){
            square =  i* i;
            printf("thread No. %d   Num : %d   square : %d\n", omp_get_thread_num(), i, square);
            sum += square;
        }
    }
    double end = omp_get_wtime();
    double total_time = end - start;

    printf("\nSum of 1-100 squares : %d", sum);
    printf("\nTime taken : %lf\n", total_time);
    return 0;
}
```

Screenshot 2:

```
C:\Users\adity\OneDrive\Documents\College\sem 7\HPC\HPC Lab\Assignment-1>a.exe
No. of Threads Allocated : 10
thread No. 7   Num : 71   square : 5041
thread No. 7   Num : 72   square : 5184
thread No. 7   Num : 73   square : 5329
thread No. 7   Num : 74   square : 5476
thread No. 7   Num : 75   square : 5625
thread No. 7   Num : 76   square : 5776
thread No. 7   Num : 77   square : 5929
thread No. 7   Num : 78   square : 6084
thread No. 7   Num : 79   square : 6241
thread No. 7   Num : 80   square : 6400
thread No. 3   Num : 31   square : 961
thread No. 3   Num : 32   square : 1024
thread No. 3   Num : 33   square : 1089
thread No. 3   Num : 34   square : 1156
```

```
thread No. 0   Num : 6   square : 36
thread No. 0   Num : 7   square : 49
thread No. 0   Num : 8   square : 64
thread No. 0   Num : 9   square : 81
thread No. 0   Num : 10   square : 100
thread No. 5   Num : 51   square : 2601
thread No. 5   Num : 52   square : 2704
thread No. 5   Num : 53   square : 2809
thread No. 5   Num : 54   square : 2916
thread No. 5   Num : 55   square : 3025
thread No. 5   Num : 56   square : 3136
thread No. 5   Num : 57   square : 3249
thread No. 5   Num : 58   square : 3364
thread No. 5   Num : 59   square : 3481
thread No. 5   Num : 60   square : 3600

Sum of 1-100 squares : 352121
Time taken : 0.177000
```

Information 2:

Once the compiler encounters the parallel regions code, threads will get created and it will calculate square of all first 100 numbers with addition of all squares of numbers stored in variable sum. Threads works in parallel and do task given to it.

Github Link :

https://github.com/Aditya-Manapure/2019BTECS00055_HPC_LAB/tree/main/Assignment-1