Name : Nisarg .k.Amlani
Roll : Ce001
Id : 22ceueg082

LAB : 07

---

1. Create a parallel version of the pi program using loop construct. Display the value of pi
and the run time and respective number of threads. (Run four instances of the program for
number of threads ranging from 1 to 4). Use following runtime library routines.
a. int omp_get_num_threads();
b. int omp_get_thread_num();
c. double omp_get_wtime();

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

static long num_steps = 10000000;
double step;

#define NUM_THREADS 4

int main()
{
   int i, nthrds;
   double pi, sum[NUM_THREADS] = {0.0};
   step = 1.0 / (double)num_steps;
   omp_set_num_threads(NUM_THREADS);
   double start = omp_get_wtime();

#pragma omp parallel
```

```
    {
        int  id, nthreads;
        double x;
        id = omp_get_thread_num();
        nthreads = omp_get_num_threads();
        if (id == 0)
        {
            nthrds = nthreads;
        }

        #pragma omp for
        for (i = 0; i < num_steps; i++)
        {
            x = (i + 0.5) * step;
            sum[id] += 4.0 / (1.0 + x * x);
        }
    }


    double end = omp_get_wtime();
    printf("nthreads: %d\n", nthrds);
    for (i = 0, pi = 0.0; i < nthrds; i++)
    {
        pi += step * sum[i];
    }


    printf("Pi with OpenMP: %.15f\n", pi);
    printf("Time taken: %.15f seconds\n", end - start);
}
```

=> Outputs

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
nthreads: 4
Pi with OpenMP: 3.141592653589670
Time taken: 0.314715906999481 seconds
```

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
nthreads: 3
Pi with OpenMP: 3.141592653589728
Time taken: 0.261631736000709 seconds
```

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
nthreads: 2
Pi with OpenMP: 3.141592653589923
Time taken: 0.204361284999322 seconds
```

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
nthreads: 1
Pi with OpenMP: 3.141592653589731
Time taken: 0.060633639999651 seconds
```

2. Write an OpenMP program to perform matrix multiplication using loop construct.

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

#define NUM_THREADS 4
#define ROW 3
#define COL 3

int main()
{
   int mat[ROW][COL] = {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}};
   int mat2[ROW][COL] = {{1, 2, 3}, {1, 2, 3}, {1, 2, 3}};
   int result[ROW][COL] = {0};

   int nthrds = 0;
   omp_set_num_threads(NUM_THREADS);
   double start = omp_get_wtime();

#pragma omp parallel
   {
       int id = omp_get_thread_num();
       int nthreads = omp_get_num_threads();

#pragma omp single
       {
           nthrds = nthreads;
       }

       // Parallelized outer and middle loops
#pragma omp for collapse(2)
       for (int i = 0; i < ROW; i++)
       {
           for (int j = 0; j < COL; j++)
           {
```

```
                int sum = 0;
                for (int k = 0; k < COL; k++)
                {
                    sum += mat[i][k] * mat2[k][j];
                }
                result[i][j] = sum;  // Safe write, each thread
writes a separate result[i][j]
            }
        }
    }

    double end = omp_get_wtime();
    printf("Threads: %d\n", nthrds);
    printf("Time Taken: %.15f seconds\n", end - start);

    printf("\nResult Matrix:\n");
    for (int i = 0; i < ROW; i++)
    {
        for (int j = 0; j < COL; j++)
        {
            printf(" %d ", result[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

=> Output

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
Threads: 4
Time Taken: 0.000181191999218 seconds

Result Matrix:
 6   12   18
 6   12   18
 6   12   18
```

3. Write an OpenMP program to compute the dot product of two vectors using the reduction clause to sum up partial results.

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

#define NUM_THREADS 4
#define num_steps 100

int main()
{
   long int prod = 0;
   int nthrds;
   int arr[num_steps], arr2[num_steps];

   for (int i = 0; i < num_steps; i++)
   {
       arr[i] = i + 1;
       arr2[i] = i + 1;
   }
```

```
    omp_set_num_threads(NUM_THREADS);
    double start = omp_get_wtime();

#pragma omp parallel
    {
        int partial_mul = 0;
        int id ;
        id = omp_get_thread_num();
        if(id == 0)
        {
            printf("Threads %d\n",omp_get_num_threads());
        }
#pragma omp for reduction(+ : prod)
        for (int i = 0; i < num_steps; i++)
        {
            partial_mul = arr[i] * arr2[i];
            prod += partial_mul;
        }
    }

    double end = omp_get_wtime();

    printf("Time Taken %.16f\n", end - start);
    printf("Dot Product %ld\n", prod);
}
```

=> Output

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
Threads 4
Time Taken 0.0002298550007254
Dot Product 338350
```

4. Write an OpenMP program to demonstrate the difference between static and dynamic scheduling in OpenMP by summing elements of a large array.

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

#define NUM_THREADS 4
#define ARR_SZ 1000000
#define CHUNK_SZ 1000

int main()
{
   int arr[ARR_SZ], sum_static = 0, sum_dynamic = 0;
   for (int i = 0; i < ARR_SZ; i++)
   {
       arr[i] = i + 1;
   }

   omp_set_num_threads(NUM_THREADS);
   double start_static = omp_get_wtime();

#pragma omp parallel
   {
       long int partial_sum = 0;

#pragma omp for schedule(static, CHUNK_SZ)
       for (int i = 0; i < ARR_SZ; i++)
       {
           partial_sum += arr[i];
       }

#pragma omp critical
       sum_static += partial_sum;
   }
```

```
    double end_static = omp_get_wtime();
    double start_dynamic = omp_get_wtime();

    omp_set_num_threads(NUM_THREADS);

#pragma omp parallel
    {
        long int partial_sum = 0;

#pragma omp for schedule(dynamic, CHUNK_SZ)
        for (int i = 0; i < ARR_SZ; i++)
        {
            partial_sum += arr[i];
        }

#pragma omp critical
        sum_dynamic += partial_sum;
    }

    double end_dynamic = omp_get_wtime();

    printf("Static Scheduling Sum: %ld, Time: %.6f\n", sum_static,
end_static - start_static);
    printf("Dynamic Scheduling Sum: %ld, Time: %.6f\n", sum_dynamic,
end_dynamic - start_dynamic);
}
```

=> Output

```
┌──(nisarg㉿fedora)-[~/…/Sem-6/Sem_6_repo/ACA/Lab-7]
└─$ ./a.out
Static Scheduling Sum: 1784293664, Time: 0.000876
Dynamic Scheduling Sum: 1784293664, Time: 0.000558
```