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

Lab : 6

Code Pi using Serial

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

static long num_steps = 10000000;
double step;
int main(){
   int i ;
   double x  , pi , sum = 0.0;

   step = 1.0/(double)num_steps;
   double start = omp_get_wtime();

   for(i =0; i<num_steps; i++)
   {
       x = (i+0.5)*step;
       sum += 4.0/(1.0+x*x);
   }
   pi = step * sum;
   double end = omp_get_wtime();
   printf("Pi with OpenMP: %.15f\n",pi);
   printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Pi with OpenMP: 3.141592653589731
Time taken: 0.022761173000617 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Pi with OpenMP: 3.141592653589731
Time taken: 0.023106741999982 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Pi with OpenMP: 3.141592653589731
Time taken: 0.023597235999659 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Pi with OpenMP: 3.141592653589731
Time taken: 0.023615467000127 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$
```

Code Pi using Parallel

```c
#include<stdlib.h>
#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];
   step = 1.0/(double) num_steps;
   omp_set_num_threads(NUM_THREADS);
   double start = omp_get_wtime();

   #pragma omp parallel
   {
```

```
        int i,id , nthreads;
        double x;
        id = omp_get_thread_num();
        nthreads = omp_get_num_threads();

        if(id == 0 ){ nthrds = nthreads;
        printf("threads: %d\n", nthreads);}
        for(i = id , sum[id] =0.0 ; i<num_steps;i = i+nthreads)
        {
            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);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Pi with OpenMP: 3.141592653589686
Time taken: 0.080148289000135 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Pi with OpenMP: 3.141592653589686
Time taken: 0.079596599000070 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Pi with OpenMP: 3.141592653589686
Time taken: 0.079434426999796 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Pi with OpenMP: 3.141592653589686
Time taken: 0.047810471000048 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$
```

Code Pi using Padding

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

static long num_steps = 10000000;
double step;

#define NUM_THREADS 4
#define PAD 8

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

   #pragma omp parallel
   {
       int i,id , nthreads;
       double x;
       id = omp_get_thread_num();
       nthreads = omp_get_num_threads();

       if(id == 0 ){ nthrds = nthreads;
       printf("threads: %d\n", nthreads);}
       for(i = id , sum[id][0] =0.0 ; i<num_steps;i = i+nthreads)
       {
           x = (i+0.5)*step;
           sum[id][0] += 4.0/(1.0+x*x);
       }
   }
   double end = omp_get_wtime();

   for(i = 0,pi = 0.0;i<nthrds;i++)
   {
```

```
        pi += step * sum[i][0];
    }
    printf("Padding Pi with OpenMP: %.15f\n",pi);
    printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Pi with OpenMP: 3.141592653589686
Time taken: 0.030534704000274 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Pi with OpenMP: 3.141592653589686
Time taken: 0.008101129999886 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Pi with OpenMP: 3.141592653589686
Time taken: 0.033630973999607 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$
```

Code Pi using Critical

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

static long num_steps = 10000000;
double step;

#define NUM_THREADS 4

int main()
```

```
{
   int i, nthrds;
   double pi;
   step = 1.0 / (double)num_steps;
   omp_set_num_threads(NUM_THREADS);
   double start = omp_get_wtime();

#pragma omp parallel
   {
       int i, id, nthreads;
       double x, sum;
       id = omp_get_thread_num();
       nthreads = omp_get_num_threads();

       if (id == 0)
       {
           nthrds = nthreads;
           printf("threads: %d\n", nthreads);
       }
       for (i = id, sum = 0.0; i < num_steps; i = i + nthreads)
       {
           x = (i + 0.5) * step;
           sum += 4.0 / (1.0 + x * x);
       }

#pragma omp critical
       pi += step * sum;
   }
   double end = omp_get_wtime();

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

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ gcc -fopenmp pi_critical.c
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
 threads: 4
 Critical Pi with OpenMP: 3.141592653589686
 Time taken: 0.007802002000062 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
 threads: 4
 Critical Pi with OpenMP: 3.141592653589686
 Time taken: 0.008252094999989 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
 threads: 4
 Critical Pi with OpenMP: 3.141592653589686
 Time taken: 0.007659092000722 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
 threads: 4
 Critical Pi with OpenMP: 3.141592653589686
 Time taken: 0.007902730999376 seconds
```

Code Pi using Atomic

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

static long num_steps = 10000000;
double step;

#define NUM_THREADS 4

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

#pragma omp parallel
```

```
    {
        int i, id, nthreads;
        double x, sum;
        id = omp_get_thread_num();
        nthreads = omp_get_num_threads();

        if (id == 0)
        {
            nthrds = nthreads;
            printf("threads: %d\n", nthreads);
        }
        for (i = id, sum = 0.0; i < num_steps; i = i + nthreads)
        {
            x = (i + 0.5) * step;
            sum += 4.0 / (1.0 + x * x);
        }
        // sum = sum *step;
#pragma atomic
        pi += step * sum;
    }
    double end = omp_get_wtime();

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

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Pi with OpenMP: 3.141592653589686
Time taken: 0.008723798000574 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Pi with OpenMP: 3.141592653589686
Time taken: 0.017979409999498 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Pi with OpenMP: 3.141592653589686
Time taken: 0.009777014999599 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Pi with OpenMP: 3.141592653589686
Time taken: 0.009044248000464 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$
```

Code Sum of Array using Serial

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

static long num_steps = 100;

int main(){
   int arr[num_steps]  , sum = 0;

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

   double start = omp_get_wtime();

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

   double end = omp_get_wtime();
   printf("Serial Sum: %d\n", sum);
   printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Serial Sum: 5050
Time taken: 0.000000424999598 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Serial Sum: 5050
Time taken: 0.000001315000191 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
Serial Sum: 5050
Time taken: 0.000000414999704 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$
```

Code Sum of Array using Parallel

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

static long num_steps = 100;
#define NUM_THREADS 4

int main()
{
   int arr[num_steps], sum[NUM_THREADS] = {0}, nthrds;

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

   omp_set_num_threads(NUM_THREADS);

   double start = omp_get_wtime();

#pragma omp parallel
   {
```

```
        int i = 0, id, nthreads;
        id = omp_get_thread_num();
        nthreads = omp_get_num_threads();
        if (id == 0)
        {
            nthrds = nthreads;
            printf("threads: %d\n", nthreads);
        }
        for (i = id; i < num_steps; i += nthreads)
        {
            sum[id] += arr[i];
        }
    }
    int total_sum = 0;
    for (int i = 0; i < nthrds; i++)
    {
        total_sum += sum[i];
    }

    double end = omp_get_wtime();
    printf("Parallel Sum: %d\n", total_sum);
    printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Sum: 5050
Time taken: 0.000288320999971 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Sum: 5050
Time taken: 0.000278491999779 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Parallel Sum: 5050
Time taken: 0.000283990999378 seconds
```

Code Sum of Array using Padding

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

static long num_steps = 100;
#define NUM_THREADS 4
#define PAD 4

int main()
{
   int arr[num_steps], sum[NUM_THREADS][PAD] = {0}, nthrds;

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

   omp_set_num_threads(NUM_THREADS);

   double start = omp_get_wtime();

#pragma omp parallel
   {
       int i = 0, id, nthreads;
       id = omp_get_thread_num();
       nthreads = omp_get_num_threads();
       if (id == 0)
       {
           nthrds = nthreads;
           printf("threads: %d\n", nthreads);
       }
       for (i = id; i < num_steps; i += nthreads)
       {
           sum[id][0] += arr[i];
       }
   }
   int total_sum = 0;
```

```
    for (int i = 0; i < nthrds; i++)
    {
        total_sum += sum[i][0];
    }

    double end = omp_get_wtime();
    printf("Padding Sum: %d\n", total_sum);
    printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Sum: 5050
Time taken: 0.000281087000076 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Sum: 5050
Time taken: 0.000274871999864 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Padding Sum: 5050
Time taken: 0.000268377000793 seconds
```

Code Sum of Array using Critical

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

static long num_steps = 100;
#define NUM_THREADS 4


int main()
{
```

```
    int arr[num_steps], sum = 0, nthrds;

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

    omp_set_num_threads(NUM_THREADS);

    double start = omp_get_wtime();

#pragma omp parallel
    {
        int i = 0, id, nthreads,partial_sum =0;
        id = omp_get_thread_num();
        nthreads = omp_get_num_threads();
        if (id == 0)
        {
            nthrds = nthreads;
            printf("threads: %d\n", nthreads);
        }
        for (i = id; i < num_steps; i += nthreads)
        {
            partial_sum += arr[i];
        }

        #pragma omp critical
        sum += partial_sum;
    }


    double end = omp_get_wtime();
    printf("Critical Sum: %d\n",sum);
    printf("Time taken: %.15f seconds\n", end - start);
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Critical Sum: 5050
Time taken: 0.000273893000667 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Critical Sum: 5050
Time taken: 0.000227043000450 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Critical Sum: 5050
Time taken: 0.000317171000461 seconds
```

Code Sum of Array using Atomic

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

static long num_steps = 100;
#define NUM_THREADS 4

int main()
{
   int arr[num_steps], sum = 0;

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

   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();

        if (id == 0)
        {
            printf("threads: %d\n", nthreads);
        }

        for (int i = id; i < num_steps; i += nthreads)
        {
#pragma omp atomic
            sum += arr[i]; // Atomic addition
        }
    }

    double end = omp_get_wtime();
    printf("Atomic Sum: %d\n", sum);
    printf("Time taken: %.15f seconds\n", end - start);

    return 0;
}
```

Output

```
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Sum: 5050
Time taken: 0.000282887000139 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Sum: 5050
Time taken: 0.000273079000181 seconds
user1@celab2-ThinkCentre-neo-50s-Gen-3:~/nisarg/lab6$ ./a.out
threads: 4
Atomic Sum: 5050
Time taken: 0.000293477999548 seconds
```