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

# LAB: 09

---

**1. Write an OpenMP program using 4 threads, each thread calculates factorial of its id and**
**then all the factorials respective to threads need to be added to get on final sum. Use shared**
**and private clauses.**
**Output: Individual threads factorial result with their respective ids and the final sum of all**
**the factorials.**

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

int main(){
   int sum = 1, fact = 1;
   omp_set_num_threads(4);
   #pragma omp parallel private(fact) reduction(*:sum)
   {
       fact = 1;
       int id = omp_get_thread_num();
       int nthrds = omp_get_num_threads();
       for(int i = 1+id; i <= 6; i+=nthrds){
           fact *=  i;
       }
       printf("ID = %d, fact = %d\n",id,fact);
       sum *= fact;
   }
   printf("Sum = %d\n",sum);
}
```

```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/lab-8/lab8]
└─$ ./a.out
ID = 2, fact = 3
ID = 0, fact = 5
ID = 3, fact = 4
ID = 1, fact = 12
Sum = 720
```

**2. Write an OpenMP program for the Pi program by making minimum changes in the serial**
**pi program that you can do.**

```c
#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();

   #pragma omp parallel for private(x) reduction(+:sum)
   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);
}
```

```
┌──(nisarg㉿fedora)-[~/…/Sem_6_repo/ACA/lab-8/lab8]
└─$ ./a.out
Pi with OpenMP: 3.141592653589811
Time taken: 0.003733335001016 seconds
```

**3. For the given C code for Mandelbrot set area computation.**
**a. Run the code multiple times and note the output.**
**b. Find the errors in the program.**
**c. Run the error free version.**

```
/*
**  PROGRAM: Mandelbrot area
**
**  PURPOSE: Program to compute the area of a  Mandelbrot set.
**           Correct answer should be around 1.510659.
**           WARNING: this program may contain errors
**
**  USAGE:   Program runs without input ... just run the executable
**
**  HISTORY: Written:  (Mark Bull, August 2011).
**           Changed "comples" to "d_comples" to avoid collsion with
**           math.h complex type (Tim Mattson, September 2011)
*/

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

```
# define NPOINTS 1000
# define MAXITER 1000

void testpoint(void);

struct d_complex{
   double r;
   double i;
};

struct d_complex c;
int numoutside = 0;

int main(){
   int i, j;
   double area, error, eps  = 1.0e-5;
   omp_lock_t lock;
   omp_init_lock(&lock);
   omp_unset_lock(&lock);

   //   Loop over grid of points in the complex plane which contains the
Mandelbrot set,
   //   testing each point to see whether it is inside or outside the set.

   #pragma omp parallel for default(shared) private(c,eps)
   for (i=0; i<NPOINTS; i++) {
       for (j=0; j<NPOINTS; j++) {
           c.r = -2.0+2.5*(double)(i)/(double)(NPOINTS)+eps;
           c.i = 1.125*(double)(j)/(double)(NPOINTS)+eps;
           omp_set_lock(&lock);
               testpoint();
           omp_unset_lock(&lock);
       }
   }

   // Calculate area of set and error estimate and output the results


area=2.0*2.5*1.125*(double)(NPOINTS*NPOINTS-numoutside)/(double)(NPOINTS*N
POINTS);
```

```
   error=area/(double)NPOINTS;

   printf("Area of Mandlebrot set = %12.8f +/- %12.8f\n",area,error);
   printf("Correct answer should be around 1.510659\n");

}

void testpoint(void){

   // Does the iteration z=z*z+c, until |z| > 2 when point is known to be
outside set
   // If loop count reaches MAXITER, point is considered to be inside the
set

   struct d_complex z;
   int iter;
   double temp;

   z=c;
   for (iter=0; iter<MAXITER; iter++){
       temp = (z.r*z.r)-(z.i*z.i)+c.r;
       z.i = z.r*z.i*2+c.i;
       z.r = temp;
       if ((z.r*z.r+z.i*z.i)>4.0) {
       numoutside++;
       break;
       }
   }

}
```

```
┌──(nisarg㉿fedora)-[~/…/ACA/lab-8/lab8/aca_lab9]
└─$ ./a.out
Area of Mandlebrot set =   5.62500000 +/-   0.00562500
Correct answer should be around 1.510659
```

**4. Create a program that executes two independent tasks in parallel using OpenMP's sections**
**Directive.**

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

void calc(int id){
   for(int i = 0 ; i < 10 ; i++)
       printf("Id = %d,  i = %d\n", id, i);
}

int main(){
   #pragma omp parallel
   {
       #pragma omp sections
       {
           #pragma omp section
               calc(omp_get_thread_num());
           #pragma omp section
               calc(omp_get_thread_num());
           #pragma omp section
               calc(omp_get_thread_num());
       }
   }
}
```

```
┌──(nisarg㉿fedora)-[~/…/ACA/lab-8/lab8/aca_lab9]
└─$ ./a.out
Id = 3,  i = 0
Id = 3,  i = 1
Id = 3,  i = 2
Id = 3,  i = 3
Id = 3,  i = 4
Id = 3,  i = 5
Id = 3,  i = 6
Id = 3,  i = 7
Id = 3,  i = 8
Id = 3,  i = 9
Id = 0,  i = 0
Id = 0,  i = 1
Id = 0,  i = 2
Id = 0,  i = 3
Id = 0,  i = 4
Id = 0,  i = 5
Id = 0,  i = 6
Id = 0,  i = 7
Id = 0,  i = 8
Id = 0,  i = 9
Id = 6,  i = 0
Id = 6,  i = 1
Id = 6,  i = 2
Id = 6,  i = 3
Id = 6,  i = 4
Id = 6,  i = 5
Id = 6,  i = 6
Id = 6,  i = 7
Id = 6,  i = 8
Id = 6,  i = 9

┌──(nisarg㉿fedora)-[~/…/ACA/lab-8/lab8/aca_lab9]
└─$
```