PARALLEL AND DISTRIBUTED COMPUTING
L – 19,20
DATE: 18.7.19
DHRUBANKA DUTTA, 17BCE1019

EXERCISES:

1. Find the sum of ‘n’ integers using the omp barrier.

CODE:

```
/*Exercise - Find the sum of 'n' integers using the omp barrier*/

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

int cum_sum (int i, int k) {
int indi_sum = 0;
for (int j=i;j<=k;j++)
indi_sum+=j;
return indi_sum;
}
int main () {
int sum = 0, n, sum1, sum2, sum3, sum4;
printf("Enter the number: ");
scanf("%d", &n);
#pragma omp parallel num_threads(4) shared(n) shared(sum) shared(sum1)
shared(sum2) shared(sum3) shared(sum4)
{
if (omp_get_thread_num() == 0)
sum1 = cum_sum(1,n/4);
else if (omp_get_thread_num() == 1)
sum2 = cum_sum(n/4+1, n/2);
else if (omp_get_thread_num() == 2)
sum3 = cum_sum(n/2+1, 3*n/4);
else if (omp_get_thread_num() == 3)
sum4 = cum_sum(3*n/4+1, n);

#pragma omp barrier
if (omp_get_thread_num() == 0) {
sum = sum1+sum2+sum3+sum4;
printf("The sum of the first n integers is %d\n", sum);
}
}
return 0;
}
```

OUTPUT :

```
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp sum_of_n_integers.c && ./a.out
Enter the number: 10
The sum of the first n integers is 55
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp sum_of_n_integers.c && ./a.out
Enter the number: 4
The sum of the first n integers is 10
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp sum_of_n_integers.c && ./a.out
Enter the number: 5
The sum of the first n integers is 15
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp sum_of_n_integers.c && ./a.out6
bash: ./a.out6: No such file or directory
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp sum_of_n_integers.c && ./a.out
Enter the number: 6
The sum of the first n integers is 21
```

2. Create 4 threads for performing the below matrix operations using omp barrier.
i. Addition
ii. Subtraction
iii. Multiplication
iv. Division

CODE:

```
/*
2. Create 4 threads for performing the below matrix operations using omp barrier.
i. Addition
ii. Subtraction
iii. Multiplication
iv. Division
*/

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

int results1[3][3], results2[3][3], results3[3][3], results4[3][3];
```

```
void addition(int a[3][3], int n) {
for (int i=0;i<3;i++)
for (int j=0;j<3;j++)
results1[i][j] = a[i][j]+n;
}

void subtraction(int a[3][3], int n) {
for (int i=0;i<3;i++)
for (int j=0;j<3;j++)
results2[i][j] = a[i][j]-n;
}

void multiplication(int a[3][3], int n) {
for (int i=0;i<3;i++)
for (int j=0;j<3;j++)
results3[i][j] = a[i][j]*n;
}

void division(int a[3][3], int n) {
for (int i=0;i<3;i++)
for (int j=0;j<3;j++)
results4[i][j] = a[i][j]/n;
}

void print_matrix (int a[3][3]) {
for (int i=0;i<3;i++) {
for (int j=0;j<3;j++)
printf("%d ", a[i][j]);
printf("\n");
}

}
int main () {
int a[3][3] = {{1,2,3}, {4,5,6}, {7,8,9}}, n=2;
#pragma omp parallel num_threads(4)
shared(a,n,results1,results2,results3,results4)
{
if (omp_get_thread_num() == 0)
addition(a,n);
else if (omp_get_thread_num() == 1)
subtraction(a,n);
else if (omp_get_thread_num() == 2)
multiplication(a,n);
else if (omp_get_thread_num() == 3)
division(a,n);

#pragma omp barrier
```

```
if ( omp_get_thread_num() == 0 ) {
printf("Matrix after addition: \n");
print_matrix(results1);
}
else if ( omp_get_thread_num() == 1 ) {
printf("Matrix after subtraction: \n");
print_matrix(results2);
}
else if ( omp_get_thread_num() == 2 ) {
printf("Matrix after multiplication: \n");
print_matrix(results3);
}
else if ( omp_get_thread_num() == 3 ) {
printf("Matrix after division: \n");
print_matrix(results4);
}
}

return 0;
}
```

OUTPUT:

```
[dhrubanka@dhrubanka-pc 17BCE1019_LAB2_PDC_18JULY]$ gcc -fopenmp matrix_ops.c &&
./a.out
Matrix after addition:
3 4 5
6 7 8
9 10 11
Matrix after subtraction:
-1 0 1
2 3 4
5 6 7
Matrix after division:
0 1 1
2 2 3
3 4 4
Matrix after multiplication:
2 4 6
8 10 12
14 16 18
```