Parallel Programming

LAB 1 - 3rd August 2018

Note: Write all programs in your observation book and record the results. Get the signature of faculty /teaching assistance.

NOTE: i) Note down the output in your observation book.

ii) Also note the observations made after adding or removing the clauses in the directives.

1.Write a C/C++ simple parallel program to display the *thread_id* and total number of threads.

Aim: To understand and analyze the working of parallel directives and the private clause.

```
/*simpleomp.c*/
#include<stdio.h>
#include<omp.h>
int main()
{
int nthreads,tid;
#pragma omp parallel private(tid)
        {
        tid=omp_get_thread_num();
        printf("Hello world from thread=%d\n",tid);
        if(tid==0)
                {
                nthreads=omp_get_num_threads();
                printf("Number of threads=%d\n",nthreads);
                }
        }
}
```

Execute the program as follows:

$gcc –o simple –fopenmp simpleomp.c

$export OMP_NUM_THREADS=2

$./simple

Number of threads in a parallel region is determined by the *if* clause, *num_threads(), omp_set_num_threads(), OMP_NUM_THREADS.*

Use these various methods to set number of threads and mention the method of setting the same.

2. Check the output of following program:

Aim : To understand the working of if clause in parallel directive..

```
/*ifparallel.c*/
#include<stdio.h>
#include<omp.h>
int main()
{
int val;
printf("Enter 0: for serial 1: for parallel\n");
scanf("%d",&val);
#pragma omp parallel if(val)
        {
        if(omp_in_parallel())
        printf("Parallel val=%d id= %d\n",val, omp_get_thread_num());
        else
        printf("Serial val=%d id= %d\n",val, omp_get_thread_num());
        }
```

```
}
```

3. Observe and record the output of following program

Aim: To understand and analyze shared clause in parallel directive.

```
 /*shared.c*/
#include<omp.h>
int main()
{
int x=0;
#pragma omp parallel shared(x)
        {
        int tid=omp_get_thread_num();
        x=x+1;
        printf(“Thread [%d]\n value of x is %d”,tid,x);
        }
}
```

4. Learn the concept of private(), firstprivate()

```
/*learn.c*/
#include<stdio.h>
#include<omp.h>
int main()
{
int i=10;
printf("Value before pragma i=%d\n",i);
#pragma omp parallel num_threads(4) private(i)
        {
```

```
        printf("Value after entering pragma i=%d tid=%d\n",i, omp_get_thread_num());
        i=i+omp_get_thread_num();   //adds thread_id to i
        printf("Value after changing value i=%d tid=%d\n",i, omp_get_thread_num());
        }
printf("Value after having pragma i=%d tid=%d\n",i, omp_get_thread_num());
}
```

*** Note down the result by changing private() to firstprivate().**

5. Demonstration of reduction clause in parallel directive.

```
#include<stdio.h>
#include<omp.h>
void main()
{
int x=0;
#pragma omp parallel num_threads(6) reduction(+:x)
        {
        int id=omp_get_thread_num();
        int threads=omp_get_num_threads();
        x=x+1;
        printf("Hi from %d\n value of x : %d\n",id,x);
        }
printf("Final x:%d\n",x);
}
```