NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL

DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 4 (Ritik Pansuriya- 181IT237)

02nd September 2020

Faculty: Dr. Geetha V and Mrs. Tanmayee

Execute following programs and put screen shots of the output. Write analysis of the result before uploading in IRIS as a single pdf file. For programming exercises, write the code and also put a screenshot of the results.

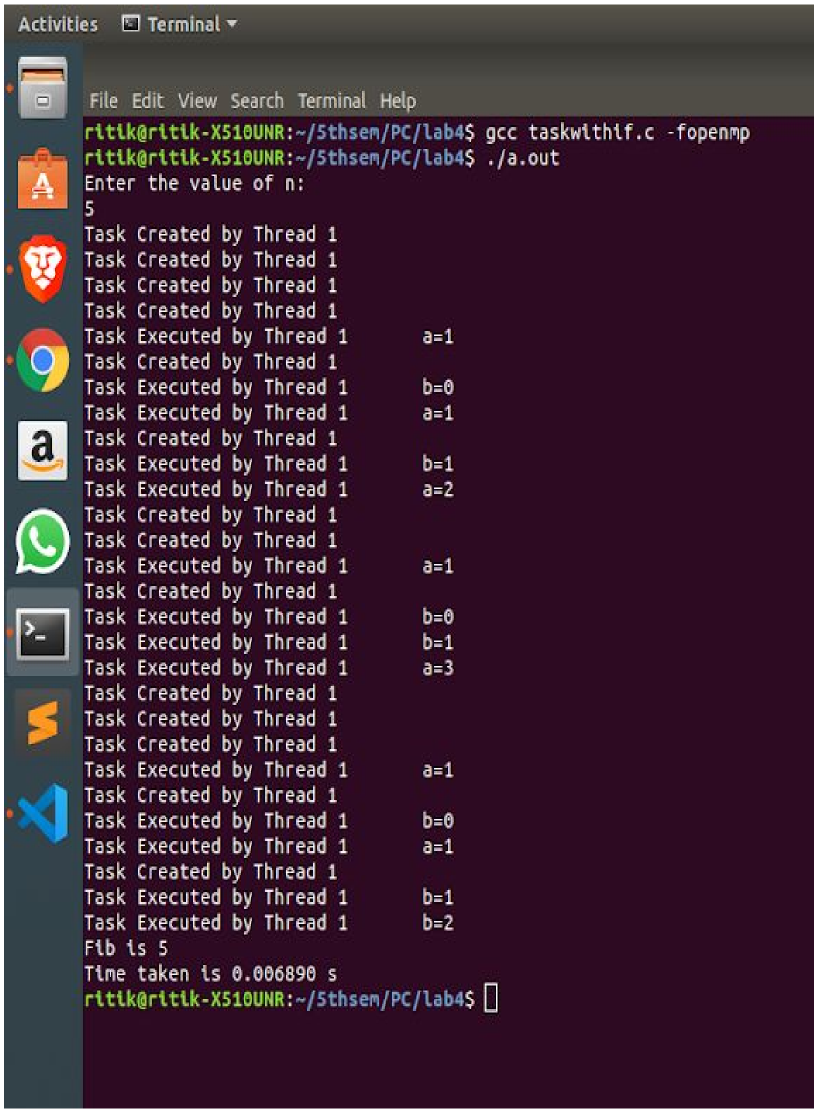
1. Program 1- Execute following code and observe the working of task directive.

Check the result by removing if() clause with task.

```
#include<stdio.h>

#include<omp.h>

int fibo(int n);

int main(void)

{

int n,fib;

double t1,t2;

printf("Enter the value of n:\n");

scanf("%d",&n);

t1=omp_get_wtime();

#pragma omp parallel shared(n)

{

#pragma omp single

{

fib=fibo(n);
}
}
t2=omp_get_wtime();
printf("Fib is %d\n",fib);
printf("Time taken is %f s \n",t2-t1);
return 0;
}
int fibo(int n)
{
int a,b;
if(n<2)
return n;
else
{
#pragma omp task shared(a) if(n>5)
{
printf("Task Created by Thread %d\n",omp_get_thread_num());
a=fibo(n-1);
printf("Task Executed by Thread %d \ta=%d\n",omp_get_thread_num(),a);
}
#pragma omp task shared(b) if(n>5)
{
printf("Task Created by Thread %d\n",omp_get_thread_num());
b=fibo(n-2);
printf("Task Executed by Thread %d \tb=%d\n",omp_get_thread_num(),b);
}
#pragma omp taskwait
```

return a+b;

}



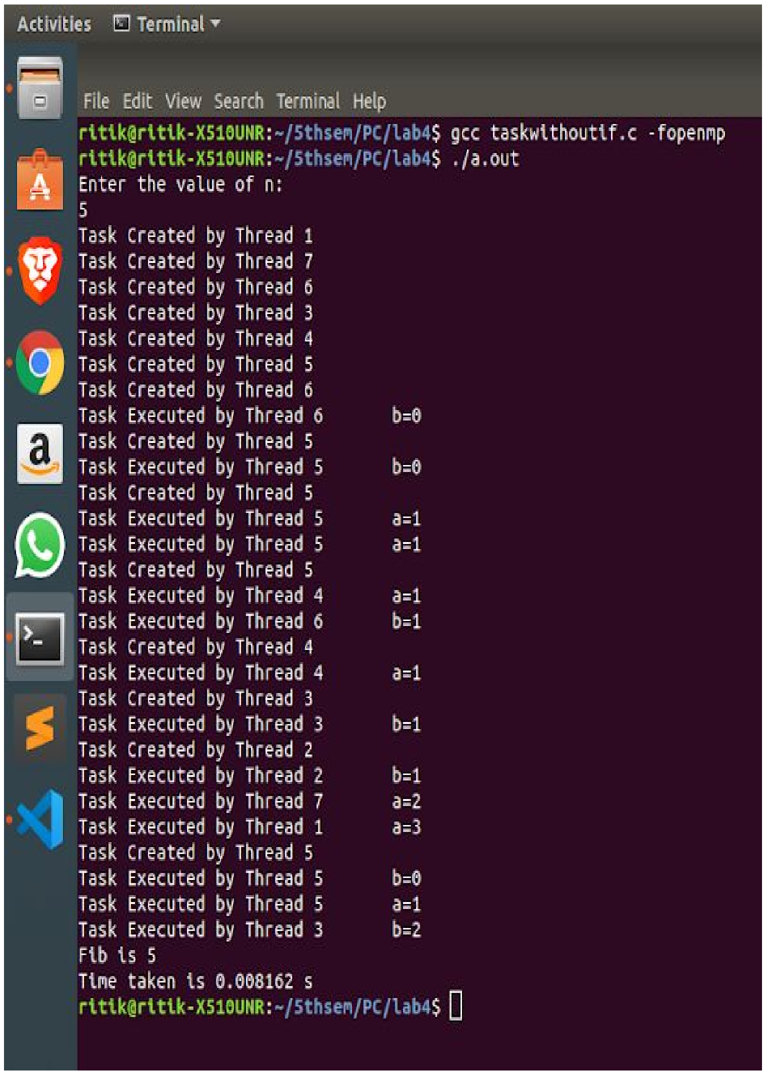
```
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ gcc taskwithif.c -fopenmp
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ ./a.out
Enter the value of n:
5
Task Created by Thread 1
Task Created by Thread 1
Task Created by Thread 1
Task Created by Thread 1
Task Executed by Thread 1       a=1
Task Created by Thread 1
Task Executed by Thread 1       b=0
Task Executed by Thread 1       a=1
Task Created by Thread 1
Task Executed by Thread 1       b=1
Task Executed by Thread 1       a=2
Task Created by Thread 1
Task Created by Thread 1
Task Executed by Thread 1       a=1
Task Created by Thread 1
Task Executed by Thread 1       b=0
Task Executed by Thread 1       b=1
Task Executed by Thread 1       a=3
Task Created by Thread 1
Task Created by Thread 1
Task Created by Thread 1
Task Executed by Thread 1       a=1
Task Created by Thread 1
Task Executed by Thread 1       b=0
Task Executed by Thread 1       a=1
Task Created by Thread 1
Task Executed by Thread 1       b=1
Task Executed by Thread 1       b=2
Fib is 5
Time taken is 0.006890 s
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ 
```

As we can see that all the execution is by one thread because of a false case in if statement(n>5) thus task will be differed in this case.

Remove if:

Now when we remove if for all the values of n tasks will be created and put in the task pool from which any thread can execute that task at any time.

Thus it gives a perfect example of parallel execution in a task.



```
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ gcc taskwithoutif.c -fopenmp
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ ./a.out
Enter the value of n:
5
Task Created by Thread 1
Task Created by Thread 7
Task Created by Thread 6
Task Created by Thread 3
Task Created by Thread 4
Task Created by Thread 5
Task Created by Thread 6
Task Executed by Thread 6       b=0
Task Created by Thread 5
Task Executed by Thread 5       b=0
Task Created by Thread 5
Task Executed by Thread 5       a=1
Task Executed by Thread 5       a=1
Task Created by Thread 5
Task Executed by Thread 4       a=1
Task Executed by Thread 6       b=1
Task Created by Thread 4
Task Executed by Thread 4       a=1
Task Created by Thread 3
Task Executed by Thread 3       b=1
Task Created by Thread 2
Task Executed by Thread 2       b=1
Task Executed by Thread 7       a=2
Task Executed by Thread 1       a=3
Task Created by Thread 5
Task Executed by Thread 5       b=0
Task Executed by Thread 5       a=1
Task Executed by Thread 3       b=2
Fib is 5
Time taken is 0.008162 s
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ 
```

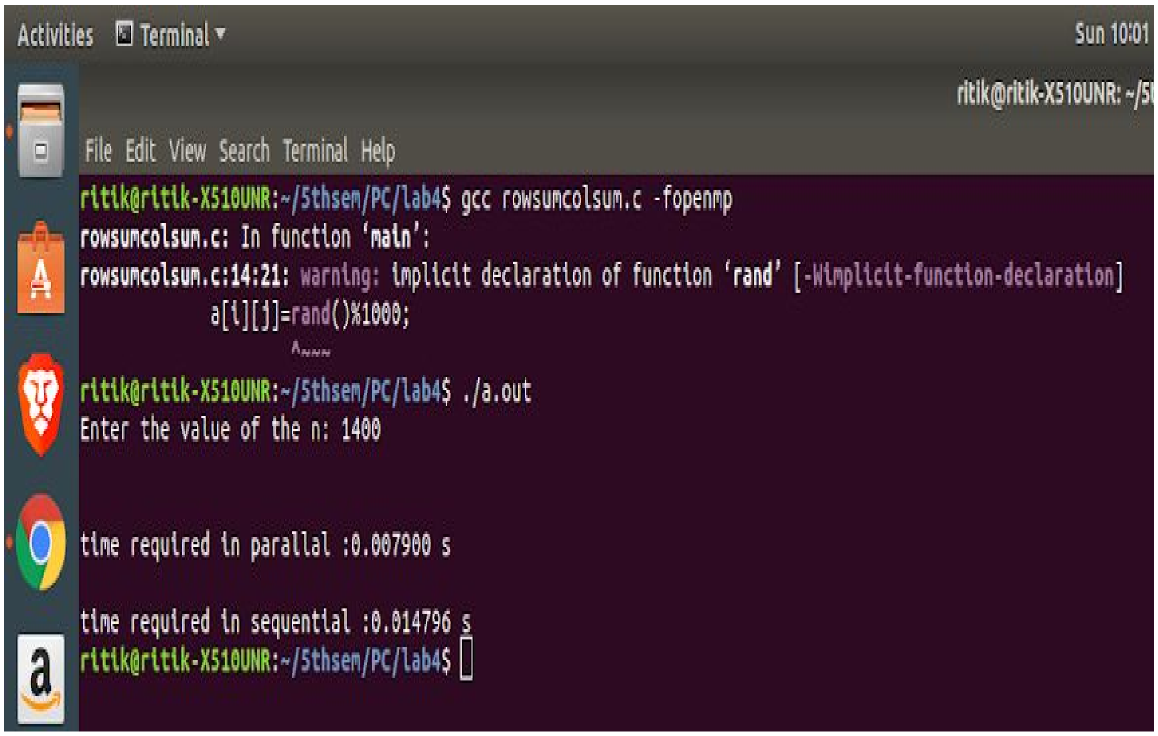
Programming exercises in OpenMP

2.Write a C/C++ OpenMP program to find ROW SUM and COLUMN SUM of a matrix a[n][n]. Compare the time of parallel execution with sequential execution.

```
#include<stdio.h>
#include<omp.h>
int main()
{
    int sum=0;
    int n;
    double t1,t2;
    printf("Enter the value of the n: ");
    scanf("%d",&n);
    int a[n][n];
    #pragma omp for schedule(static,8) collapse(2)
    for(int i=0;i<n;i++){
        for(int j=0; j<n; j++){
            a[i][j]=rand()%1000;
        }
    }
    // n=3;
    // int a[3][3]={
    //     1,2,3,
    //     4,5,6,
    //     7,8,9
    // };
    t1=omp_get_wtime();
    // printf("Row sum is : ");
```

```
for(int i=0; i<n; i++){
    sum=0;
    #pragma omp parallel for schedule(static,8) reduction(+:sum)
    for(int j=0; j<n; j++) sum=sum+a[i][j];
    // printf("%d ",sum);
}
// printf("\n\nCol sum is : ");
for(int i=0; i<n; i++){
    sum=0;
    #pragma omp parallel for schedule(static,8) reduction(+:sum)
    for(int j=0; j<n; j++) sum=sum+a[j][i];
    // printf("%d ",sum);
}
t2=omp_get_wtime();
printf("\n\ntime required in parallal :%f s",t2-t1);
t1=omp_get_wtime();
// printf("\n\nRow sum is : ");
for(int i=0; i<n; i++){
    sum=0;
    for(int j=0; j<n; j++) sum=sum+a[i][j];
    // printf("%d ",sum);
}
// printf("\n\nCol sum is : ");
for(int i=0; i<n; i++){
    sum=0;
    for(int j=0; j<n; j++) sum=sum+a[j][i];
    // printf("%d ",sum);
    }
    t2=omp_get_wtime();
    printf("\n\ntime required in sequential :%f s\n",t2-t1);
    return 0;
}
```

o/p:



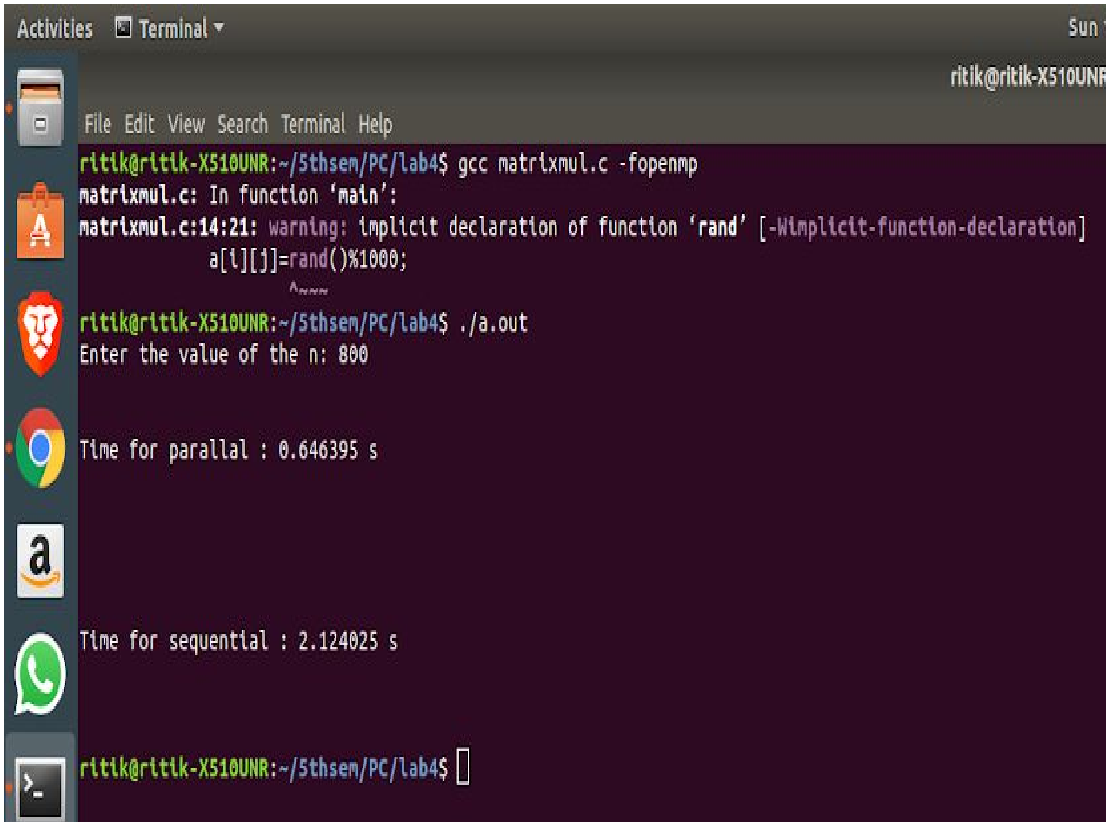
```
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ gcc rowsumcolsum.c -fopenmp
rowsumcolsum.c: In function ‘main’:
rowsumcolsum.c:14:21: warning: implicit declaration of function ‘rand’ [-Wimplicit-function-declaration]
             a[i][j]=rand()%1000;
                     ^~~~
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ ./a.out
Enter the value of the n: 1400


time required in parallal :0.007900 s

time required in sequential :0.014796 s
ritik@ritik-X510UNR:~/5thsem/PC/lab4$
```

Here i implemented the code by giving parallel execution for all the different values of rows which gives us least runtime at n=1400 in row sum and for all different columns which gives us least runtime at n=1400 in col sum then i wrote the code for sequential execution and then i compared the value for both. I wrote “”” #pragma omp parallel for schedule(static,8) reduction(+:sum) “””

3. Write a C/C++ OpenMP program to perform matrix multiplication. Compare the time of parallel execution with sequential execution.

```
#include <stdio.h>

#include <omp.h>

int main()

{

    int n;

    double t1,t2;

    printf("Enter the value of the n: ");

    scanf("%d",&n);

    int a[n][n],b[n][n];

    int c[n][n];

    #pragma omp for schedule(static,8) collapse(2)

    for(int i=0;i<n;i++){

        for(int j=0; j<n; j++){

            a[i][j]=rand()%1000;

            b[i][j]=rand()%1000;

        }

    }

    t1=omp_get_wtime();

    #pragma omp parallel for schedule(static,12) num_threads(12)

    for(int i=0; i<n; i++){

        for(int j=0; j<n; j++){

            for(int k=0; k<n; k++){

                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    t2=omp_get_wtime();
    printf("\n\nTime for parallal : %f s\n\n\n\n", t2-t1);
    // for (int i=0; i<n; i++){
    //     for (int j=0; j<n; j++) printf("%d ",c[i][j]);
    //     printf("\n");
    // }
    for (int i=0; i<n; i++){
        for (int j=0; j<n; j++) c[i][j]=0;
    }
    t1=omp_get_wtime();
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            for(int k=0; k<n; k++){
                c[i][j]+=a[i][k]*b[k][j];
            }
        }
    }
    t2=omp_get_wtime();
    printf("\n\nTime for sequential : %f s\n\n\n\n", t2-t1);
    // for (int i=0; i<n; i++){
    //     for (int j=0; j<n; j++) printf("%d ",c[i][j]);
    //     printf("\n");
    // }
}
```

o/p:



```
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ gcc matrixmul.c -fopenmp
matrixmul.c: In function ‘main’:
matrixmul.c:14:21: warning: implicit declaration of function ‘rand’ [-Wimplicit-function-declaration]
            a[i][j]=rand()%1000;
                    ^~~~
ritik@ritik-X510UNR:~/5thsem/PC/lab4$ ./a.out
Enter the value of the n: 800


Time for parallal : 0.646395 s





Time for sequential : 2.124025 s



ritik@ritik-X510UNR:~/5thsem/PC/lab4$ 
```

Here I implemented the code with 3 for loops algorithm for matrix multiplication calculation. In parallel i used “”” #pragma omp parallel for schedule(static,12) num_threads(12) ””” with static and chunk size 12 and number of threads as 12 which gives me least runtime in parallel execution. Here I gave n=800 for running the program.