NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA SURATHKAL
DEPARTMENT OF INFORMATION TECHNOLOGY

IT 301 Parallel Computing LAB 4

02nd September 2020

Faculty: Dr. Geetha V and Mrs. Tanmayee

Name: Bhagyashri Nilesh Bhmare

Roll No.: 181IT111

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 screenshot of the results.

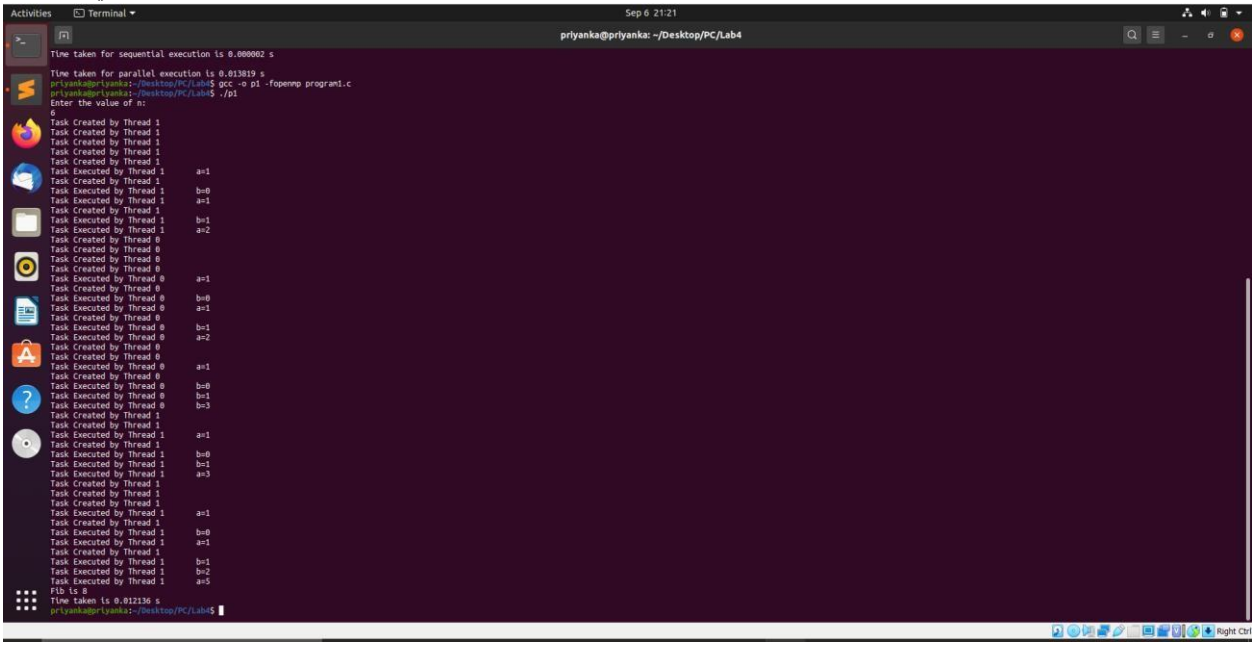
1. Program 1

Execute following code and observe the working of task directive. Check the result by removing if() clause with task.

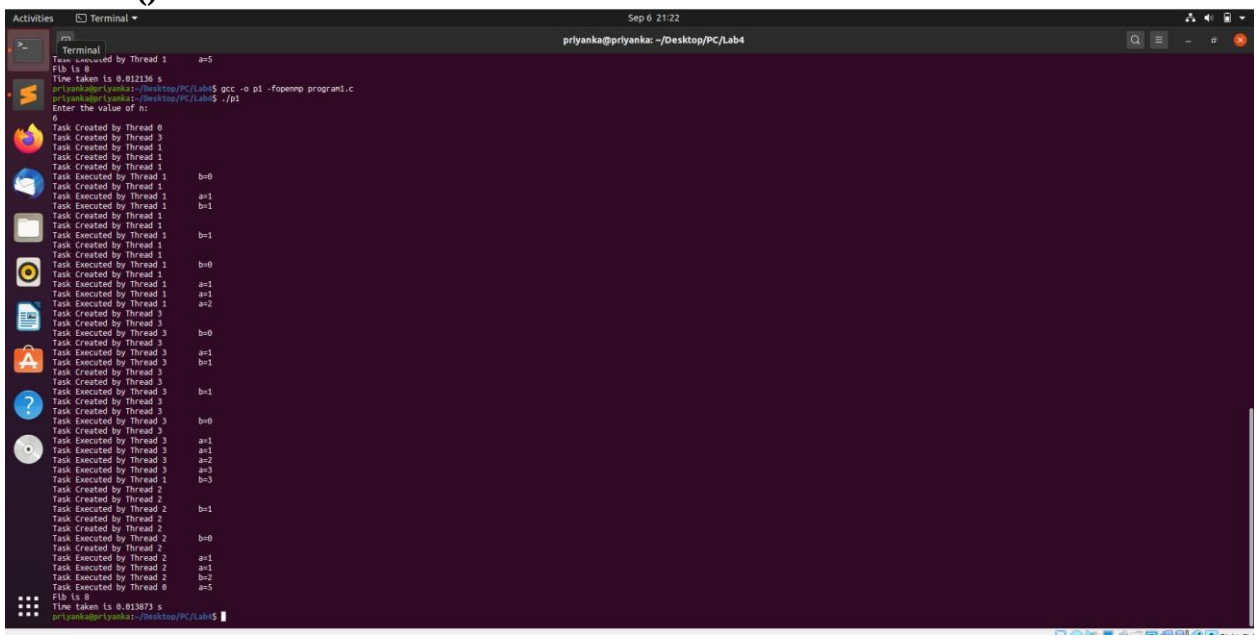
```
#include<stdio.h>
#include<omp.h>
int fibo(int n); int
main(void)
{ int n,fib;
double t1,t2;
printf("Enter the value of n:\n");
scanf("%d",&n);
t1=omp_get_wtime(); #pragma
omp parallel shared(n)
{
#pragma omp single
{
fib=fibo(n);
}
}
t2=omp_get_wtime(); printf("Fib is
%d\n",fib); printf("Time taken 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;
}
```

With if() clause:



Without if() clause:



Task Scheduling if done for all values of n when if clause in not used, while when if clause is used then is it not done for all values of n

Programming exercises in OpenMP

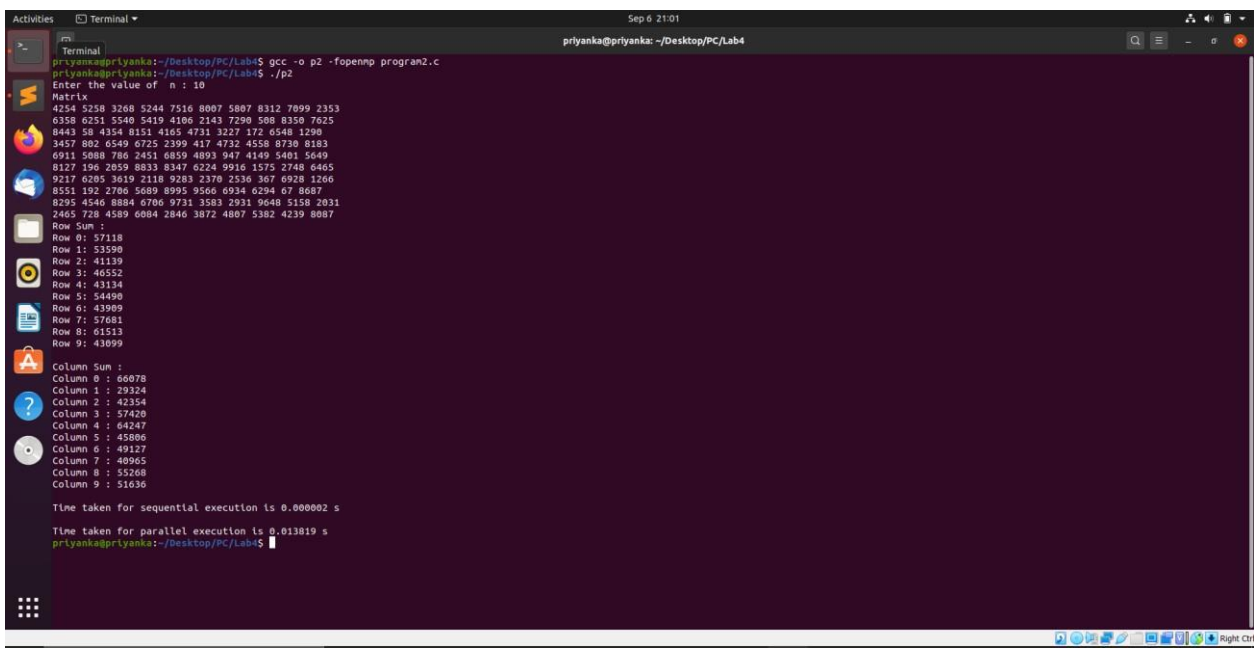
2.Write a C/C++ OpenMP program to find ROWSUM and COLUMNSUM of a matrix a[n][n]. Compare the time of parallel execution with sequential execution.

Program:

```
#include <stdio.h>
#include <omp.h>
#include <stdlib.h> #include
<time.h>
int main(void)
{
   int n, i, j;
   double     t1_s,     t2_s,     t1_p,     t2_p;
printf("Enter the value of  n : ");
   scanf("%u",   &n);
int  m[n][n];          int
row_sum[n];          int
column_sum[n];
srand(time(0));
```

```
printf("Matrix\n");
for (i = 0; i < n; i++)
   {
      for (j = 0; j < n; j++)
      {
         int k = (rand()) % 10000;
         m[i][j]          =          k;
printf("%d ", k);
      }
      printf("\n");
   }
   //sequential          t1_s  =
omp_get_wtime();      for (i
= 0; i < n; i++)
   {
      int temp = 0;
for (j = 0; j < n; j++)
temp += m[i][j];
      row_sum[i] = temp;
   }
   for (i = 0; i < n; i++)
   {
      int temp = 0;        for (j
= 0; j < n; j++)             temp
+=                          m[j][i];
column_sum[i] = temp;
   }
   t2_s = omp_get_wtime();
   int temp;

   t1_p = omp_get_wtime(); #pragma
omp parallel shared(n)
   {
#pragma omp for schedule(static, 10) private(i, j, temp)
for (i = 0; i < n; i++)
      {                 temp = 0;
for (j = 0; j < n; j++)
temp        +=        m[i][j];
row_sum[i] = temp;
      }
#pragma omp for schedule(static, 10) private(i, j, temp)
for (i = 0; i < n; i++)
      {                 temp = 0;
for (j = 0; j < n; j++)
```

```
temp += m[j][i];
column_sum[i] = temp;
    }
  }
  t2_p = omp_get_wtime();
  //results
  printf("Row Sum : \n");
for (i = 0; i < n; i++)
  {
    printf("Row %d: %d \n", i, row_sum[i]);
  }
  printf("\nColumn Sum : \n");
for (i = 0; i < n; i++)
  {
    printf("Column %d : %d \n", i, column_sum[i]);
  }
  printf("\nTime taken for sequential execution is %f s \n", t2_s - t1_s);
printf("\nTime taken for parallel execution is %f s \n", t2_p - t1_p);  return
0;
}
```



The sequential takes less time than the parallel

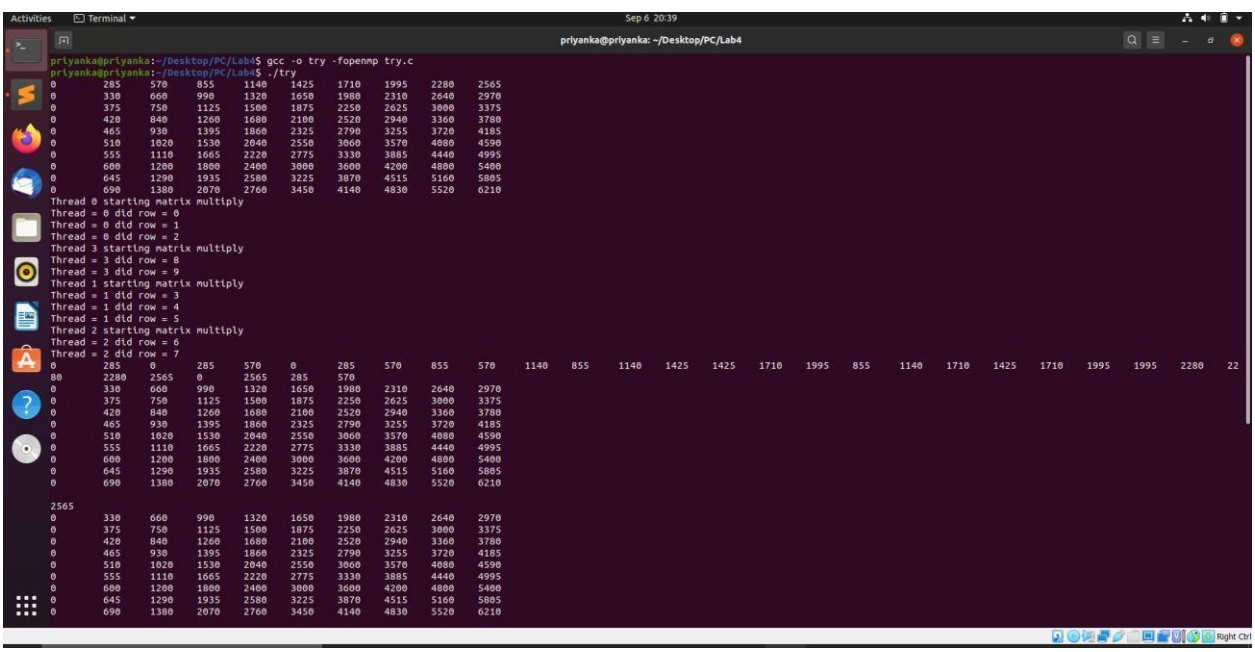
3. Write a C/C++ OpenMP program to perform matrix multiplication. Compare the time of parallel execution with sequential execution.

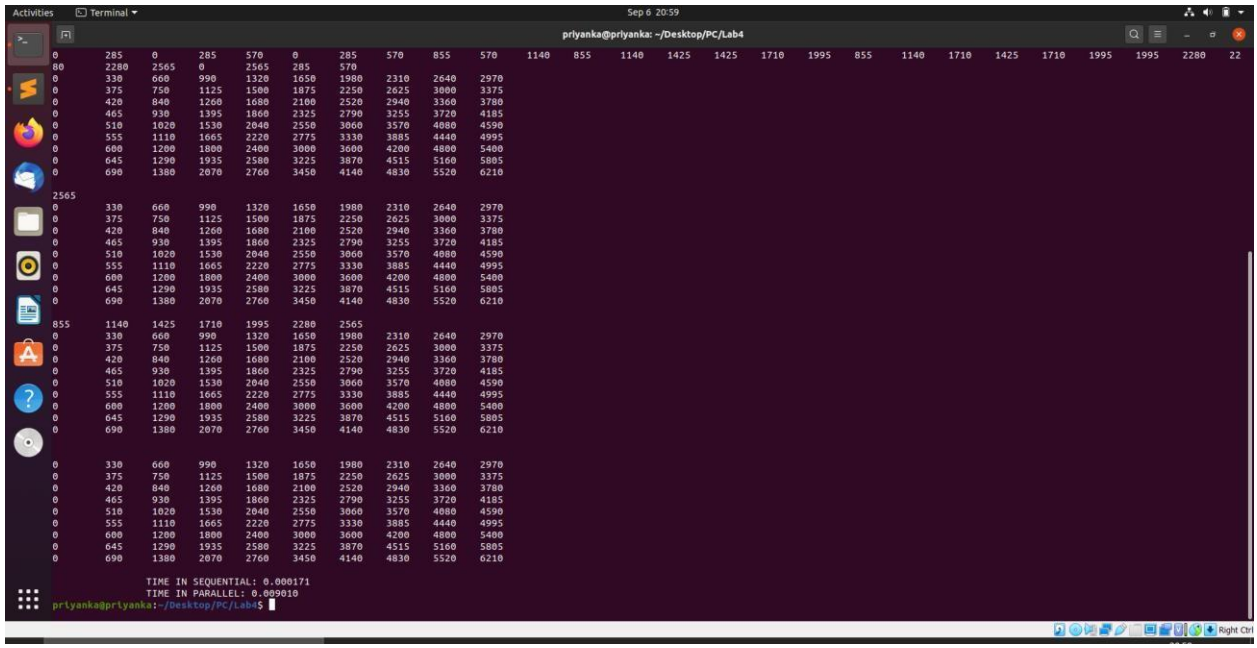
Program:

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

#define N 10 //number of rows in matrix A
#define P 10 //number of columns in matrix B
#define M 10 //number of columns in matrix A

int main(int argc, char *argv[])
{
   int tid, nthreads, i, j, k;      int
a[N][P], b[P][M], c[N][M];

   clock_t beginS = clock();
/* code for sequential */
for (int i = 0; i < N; i++)
   {
      for (int j = 0; j < P; j++)
      {
         a[i][j] = i + j;
      }
   }

   for (int i = 0; i < P; i++)
   {
      for (int j = 0; j < M; j++)
      {
         b[i][j] = i * j;
      }
   }

   for (int i = 0; i < N; i++)
   {
      for (int j = 0; j < M; j++)
      {          c[i][j]
= 0;
      }
   }
```

```
   for (i = 0; i < N; i++)
   {
      for (j = 0; j < M; j++)
         for  (k  =  0;  k  <  P;  k++)
c[i][j] += a[i][k] * b[k][j];
   }

   for (i = 0; i < N; i++)
      {
         for  (j  =  0;  j  <  M;  j++)
printf("%d\t        ",        c[i][j]);
printf("\n");
      }

   clock_t endS = clock();
   double timeS = (double)(endS - beginS) / CLOCKS_PER_SEC;


   clock_t begin = clock();

#pragma omp parallel shared(a, b, c, nthreads) private(tid, i, j, k)
   {
      tid = omp_get_thread_num();

//initializing matrix a with values i+j
#pragma omp for
      for (i = 0; i < N; i++)
for (j = 0; j < P; j++)
a[i][j] = i + j;
//initializing matrix b with values i*j
#pragma omp for
      for  (i  =  0;  i  <  P;  i++)
for (j = 0; j < M; j++)
            b[i][j] = i * j;
//initializing matrix c with values 0
#pragma omp for
      for  (i  =  0;  i  <  N;  i++)
for  (j  =  0;  j  <  M;  j++)
c[i][j] = 0;

      printf("Thread %d starting matrix multiply\n", tid);
#pragma omp for
      for (i = 0; i < N; i++)
```

```
        {
            printf("Thread = %d did row = %d\n", tid, i);
            for  (j  =  0;  j  <  M;  j++)
for (k = 0; k < P; k++)
                c[i][j]  +=  a[i][k]  *  b[k][j];
}

        for (i = 0; i < N; i++)
        {
            for  (j  =  0;  j  <  M;  j++)
printf("%d\t        ",        c[i][j]);
printf("\n");
        }
        printf("\n");
    }
    clock_t end = clock();
    double time_spent = (double)(end - begin) / CLOCKS_PER_SEC;

        printf("\t\tTIME        IN        SEQUENTIAL:        %f\n",        timeS);
printf("\t\tTIME IN PARALLEL: %f\n", time_spent);
}
```





The sequential takes less time than the parallel