HPC LAB

ASSIGNMENT NO.1

Name: Akhilesh Kumar Pal
Prn No.2019BTECS00097
Batch:B3

Q1) Write a program to print Hello World using OpenMp

code:

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

int main()
{
        #pragma omp parallel
        {

                int ID = omp_get_thread_num();
                printf("Hello from thread %d\n",ID);

        }
   return 0;
}
```
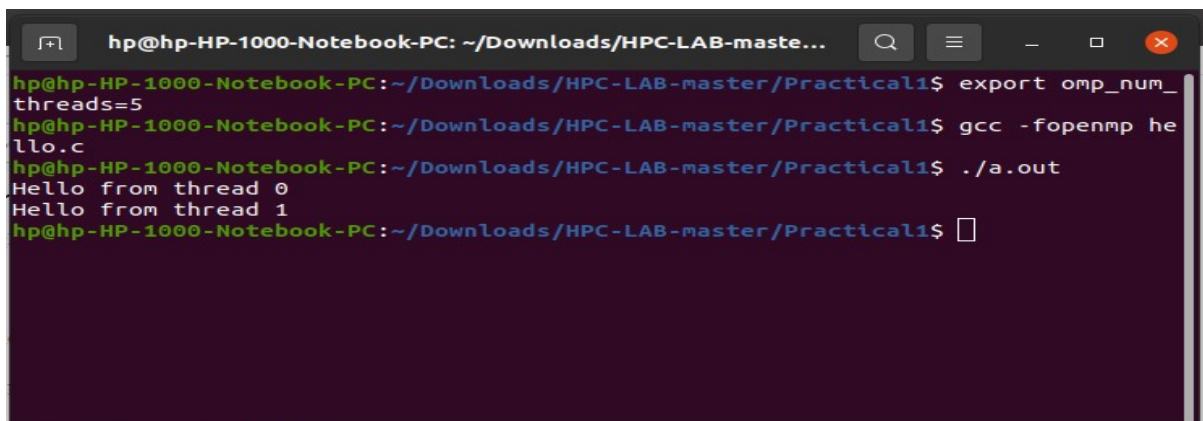
output:

Q 2)

Write a program to print the sum of squares of numbers from 1 to hundred using OpenMp.

Code:

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

```cpp
#include<stdlib.h>
#include<iostream>
using namespace std;
static int sum =0;
int main()
{
    #pragma omp parallel
    {

        for(int i=1; i<=100;i++)
        {
            if(i%4==omp_get_thread_num())
            {
                printf("thread No. %d Number : %d Square : %d\n", omp_get_thread_num(), i, i * i);
                sum+=i*i;
                printf("Sum is %d ",sum);
                cout<<endl;


            }
        }
    }
    return 0;
}
```

output:

Q3)
Write a

programme to calculate the 5the speedup in parallel and sequential execution

code: parallel execution

```cpp
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>
#include<bits/stdc++.h>
using namespace std;
```

```
int main(){
        long long sum = 0;

        double getInTime = omp_get_wtime();

        #pragma omp parallel for reduction(+ : sum)

        for(int i=1;i<=100000000;i++){
                sum += (i*i);
         }

        double getOutTime = omp_get_wtime();

        double exptTime = getOutTime - getInTime;


        printf("Time Required For Execution in Parallel : %f\n",exptTime);

        printf("Answer is : %lld",sum);

        return 0;

}
```
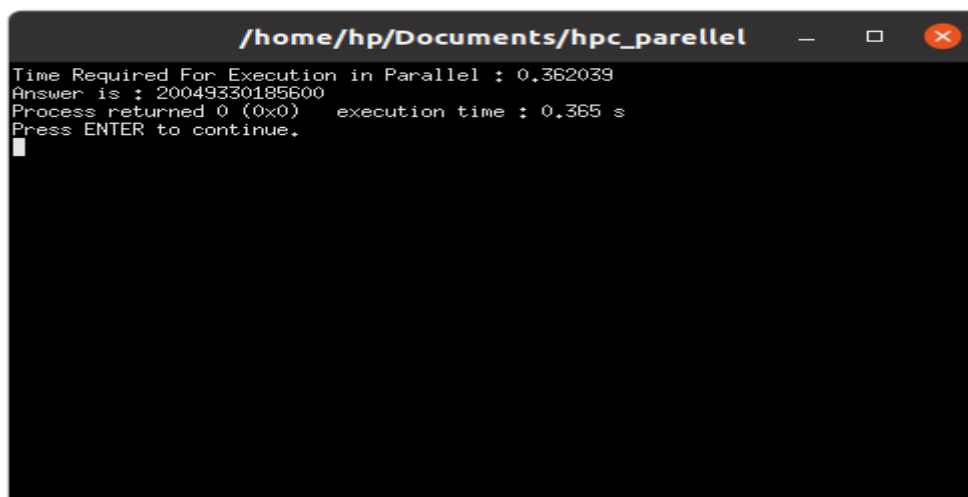
output:

code Sequential Execution:

```
#include<omp.h>
#include<stdio.h>
#include<stdlib.h>
#include<bits/stdc++.h>
using namespace std;
```

```
int main(){

      long long sum = 0;

      double inTime = omp_get_wtime();


      int i;
      for(i=1;i<=100000000;i++){
            sum += (i*i);
       }


       double outTime = omp_get_wtime();

       double expcTime = outTime - inTime;

       printf("Time Required for Execution in Serial : %f\n",expcTime);
       printf("Answer is : %lld",sum);

       return 0;
}
```
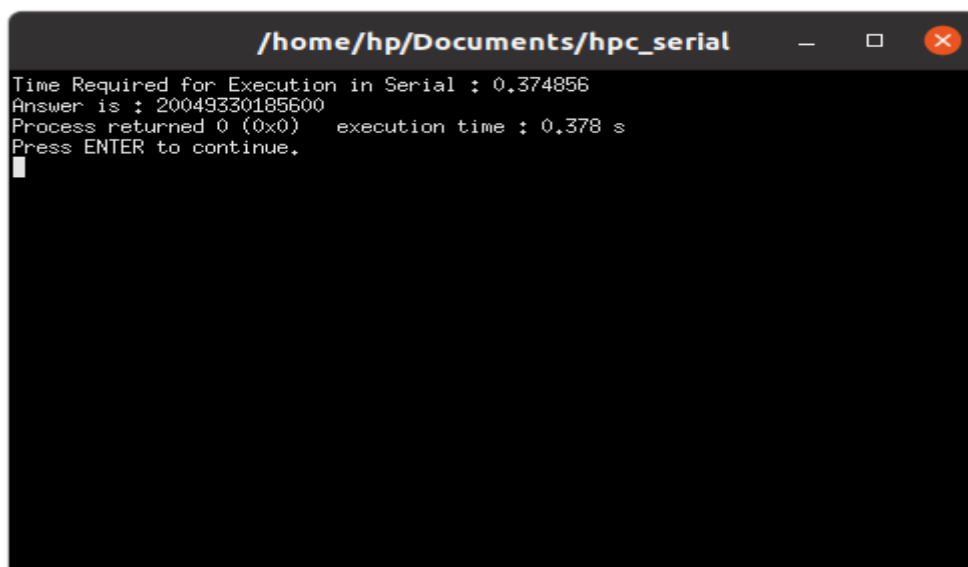
output:

Hence ,

 since speedup=sequential execution / parallel execution
  so
                =0.37486/0.362039

                =1.03541331183