Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2024-25 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 2

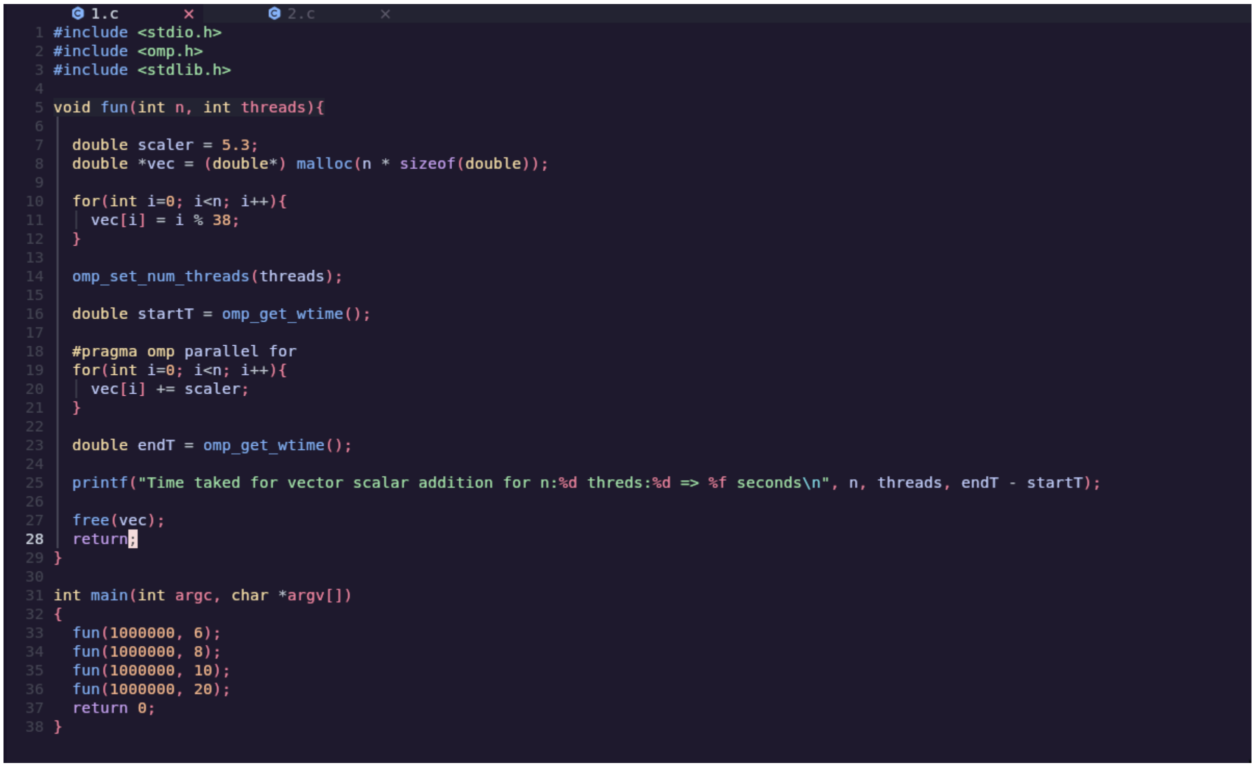
Exam Seat No: 21510111

Title of practical: Study and implementation of basic OpenMP clauses

Implement following Programs using OpenMP with C:

1. Vector Scalar Addition
2. Calculation of value of Pi
   Analyse the performance of your programs for different number of threads and Data size.

Problem Statement 1: Vector Scalar Addition
Screenshots:

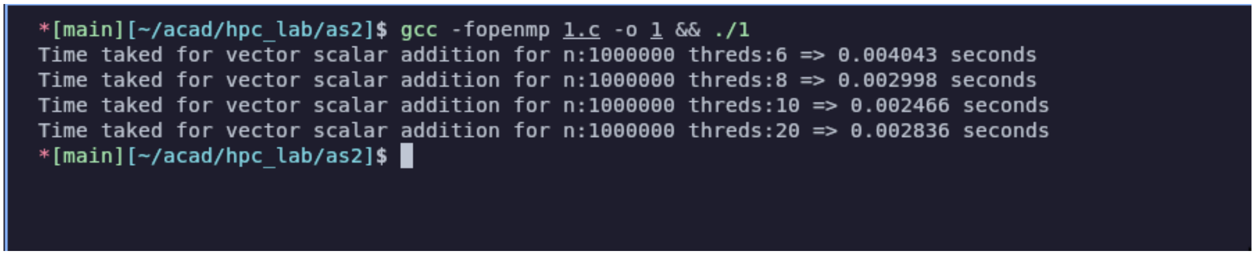


```c
#include <stdio.h>
#include <omp.h>
#include <stdlib.h>

void fun(int n, int threads){

  double scaler = 5.3;
  double *vec = (double*) malloc(n * sizeof(double));

  for(int i=0; i<n; i++){
    vec[i] = i % 38;
  }

  omp_set_num_threads(threads);

  double startT = omp_get_wtime();

  #pragma omp parallel for
  for(int i=0; i<n; i++){
    vec[i] += scaler;
  }

  double endT = omp_get_wtime();

  printf("Time taked for vector scalar addition for n:%d threds:%d => %f seconds\n", n, threads, endT - startT);

  free(vec);
  return;
}

int main(int argc, char *argv[])
{
  fun(1000000, 6);
  fun(1000000, 8);
  fun(1000000, 10);
  fun(1000000, 20);
  return 0;
}
```

Analysis:

To analyze the performance, we used different size of vectors and different number of threads.

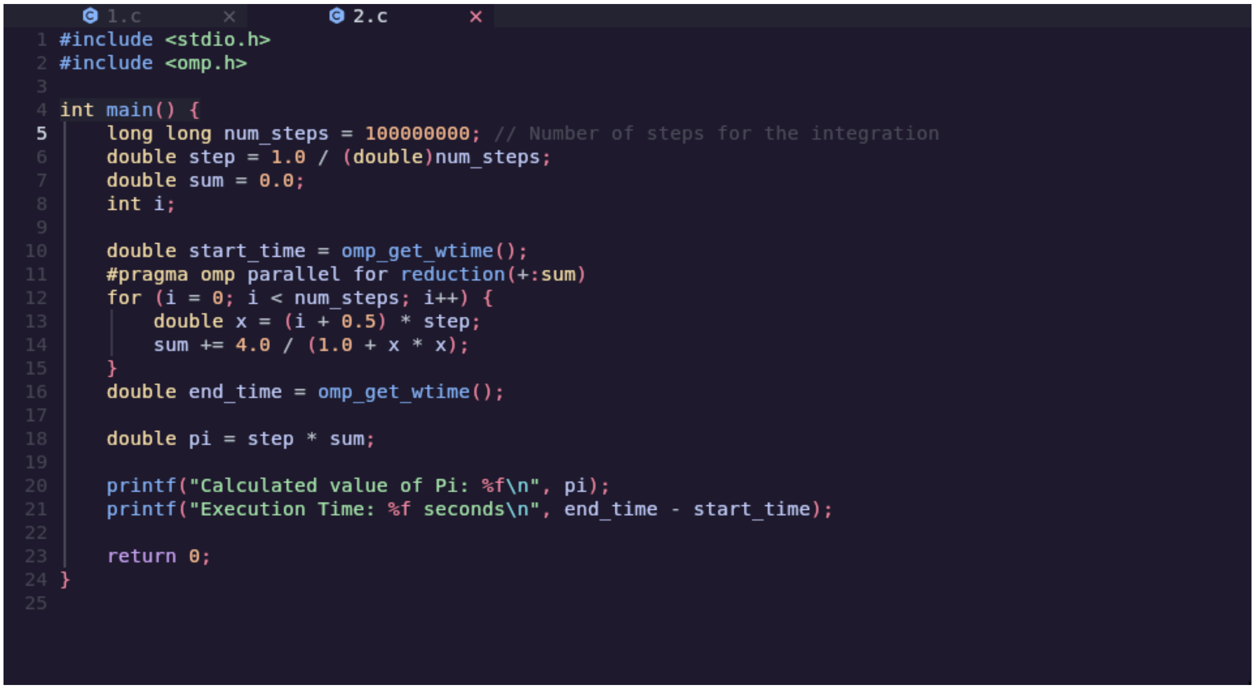
Increasing the number of threads increases the performance and decreases the time taken for operation.



```
*[main][~/acad/hpc_lab/as2]$ gcc -fopenmp 1.c -o 1 && ./1
Time taked for vector scalar addition for n:1000000 threds:6 => 0.004043 seconds
Time taked for vector scalar addition for n:1000000 threds:8 => 0.002998 seconds
Time taked for vector scalar addition for n:1000000 threds:10 => 0.002466 seconds
Time taked for vector scalar addition for n:1000000 threds:20 => 0.002836 seconds
*[main][~/acad/hpc_lab/as2]$ 
```

Problem Statement 2: Calculation of value of Pi

Screenshots:

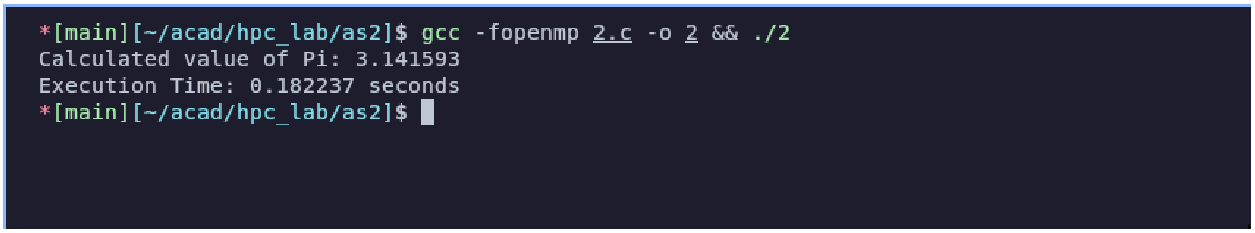


```c
#include <stdio.h>
#include <omp.h>

int main() {
    long long num_steps = 100000000; // Number of steps for the integration
    double step = 1.0 / (double)num_steps;
    double sum = 0.0;
    int i;

    double start_time = omp_get_wtime();
    #pragma omp parallel for reduction(+:sum)
    for (i = 0; i < num_steps; i++) {
        double x = (i + 0.5) * step;
        sum += 4.0 / (1.0 + x * x);
    }
    double end_time = omp_get_wtime();

    double pi = step * sum;

    printf("Calculated value of Pi: %f\n", pi);
    printf("Execution Time: %f seconds\n", end_time - start_time);

    return 0;
}
```

Information:

Calculating Pi is through numerical integration using the Monte Carlo method or more commonly, using Riemann sums (numerical integration).

Analysis:



```
*[main][~/acad/hpc_lab/as2]$ gcc -fopenmp 2.c -o 2 && ./2
Calculated value of Pi: 3.141593
Execution Time: 0.182237 seconds
*[main][~/acad/hpc_lab/as2]$
```

Github Link: https://github.com/Sidd-77/hpc-lab/tree/main/as2