Final Year B.Tech. (CSE) – VII [2024-25]

**6CS452: High Performance Computing Lab**

Assignment No: 2

# Date: 12/08/2024

**Study and implementation of basic OpenMP clauses**

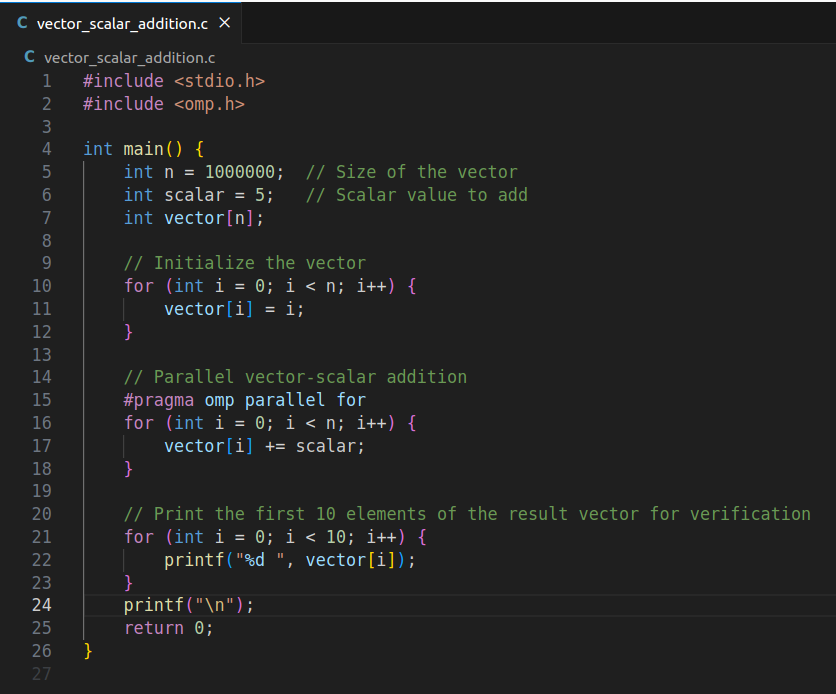
**PRN:** 21510017  **Name:** Onkar Anand Yemul

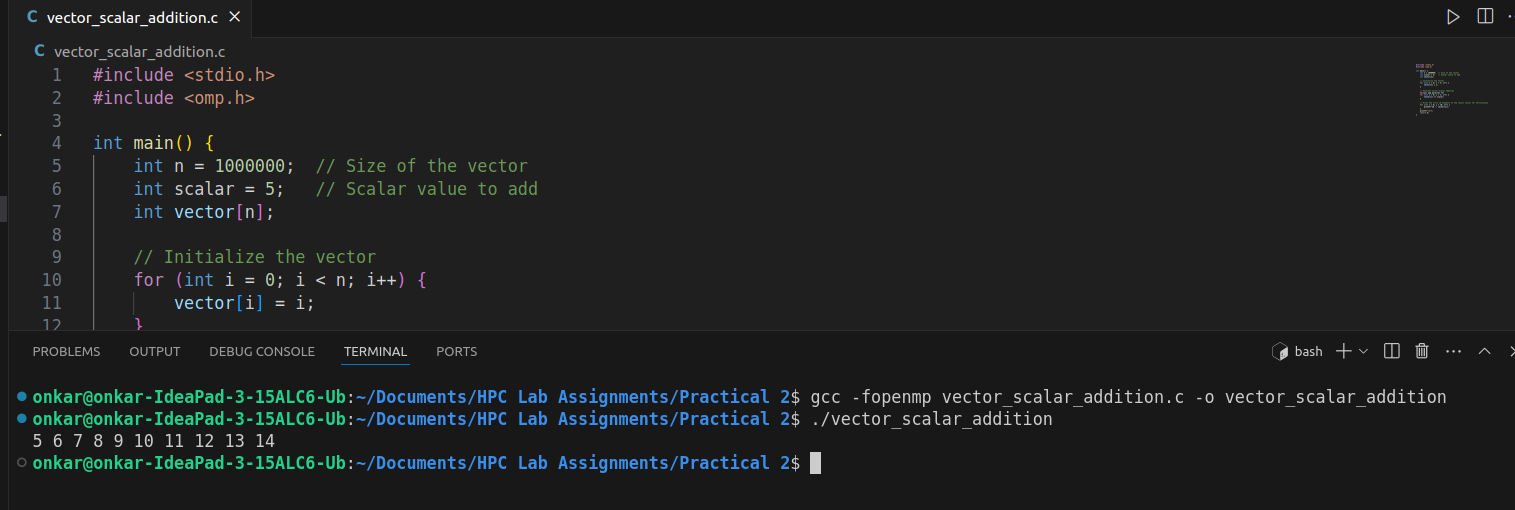
**Title:** Study and implementation of basic OpenMP clauses

**Implement following Programs using OpenMP with C and analyse the performance of your programs for different number of threads and Data size.**

**Problem Statement 1: Vector Scalar Addition**

**Screenshots:**

****



**Information:**

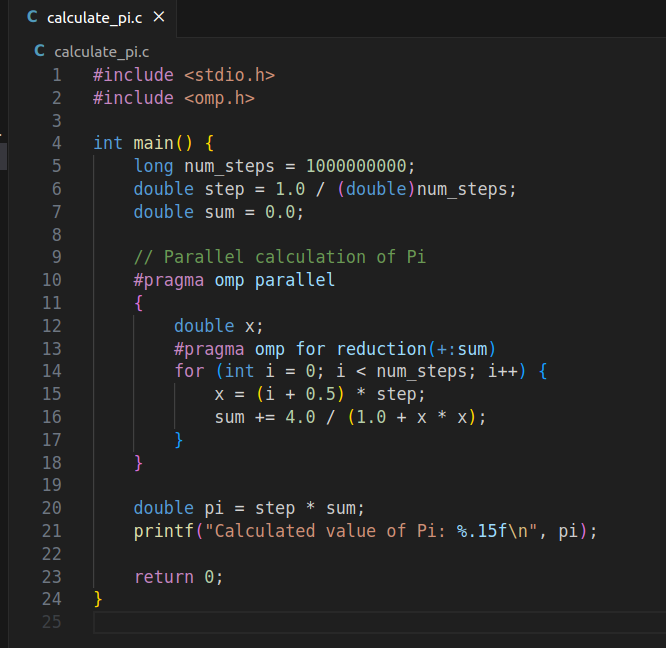
This program demonstrates the addition of a scalar value to each element of a vector using OpenMP to parallelize the operation. The vector is initialized with sequential values, and each element is incremented by the scalar value.

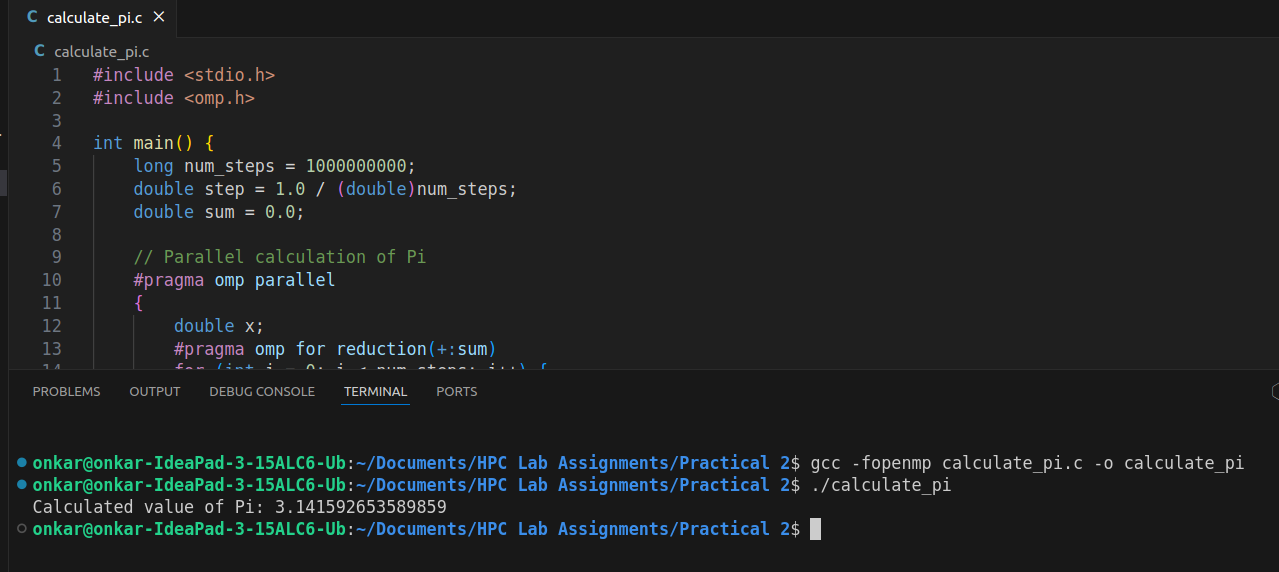
**Analysis:**

* Performance Analysis: The performance of the vector-scalar addition operation can be analyzed by varying the number of threads used. For instance, you can run the program with 1, 2, 4, and 8 threads and compare the execution times.
* Thread Scaling: The execution time should decrease as the number of threads increases, up to a certain point. However, due to overhead and limited parallelism, the performance gain might diminish or even worsen with too many threads.

**Problem Statement 2:** Calculation of value of Pi

**Screenshots:**





**Information:**

This program calculates the value of Pi using numerical integration. The integral of 4/(1 + x^2) from 0 to 1 gives the value of Pi. OpenMP is used to parallelize the summation of the series.

**Analysis:**

* **Performance Analysis:** Similar to the vector-scalar addition, the performance of the Pi calculation can be analyzed by varying the number of threads and observing the effect on execution time.
* **Accuracy and Convergence:** Increasing the number of steps improves the accuracy of Pi but also increases computation time. The optimal balance between accuracy and performance should be considered, especially when varying the number of threads.

**Github Link:**