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

**Class:** Final Year B.Tech(Computer Science and Engineering)

**Year:** 2025-26 **Semester:** 1

**Course:** High Performance Computing Lab

**PRN : 22510021** **Batch : B7**

**Practical No. 2**

**Exam Seat No:**

**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:**

1. **Code:**

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


int main(){

    int n;
    printf("Enter desired number of threads \n");
    scanf("%d", &n);

    int arr1[5] = {3, 6, 9, 11, 21};
    int arr2[5] = {21, 11, 9, 6, 3};
    int res[5];

    omp_set_num_threads(n);

    #pragma omp parallel for
    for(int i=0; i<5; i++){
```

```
        res[i] = arr1[i] + arr2[i];
        printf(" addition of %dth elements is done by thread number
%d \n", i, omp_get_thread_num());
    }


    printf(" Result array is " );

    for(int i=0; i<5; i++){
        printf("%d ", res[i]);
    }
}
```

2. **Result:**

```
PS C:\Lab\HPC\2> .\vectoradd
Enter desired number of threads
5
 addition of 1th elements is done by thread number 1
 addition of 0th elements is done by thread number 0
 addition of 3th elements is done by thread number 3
 addition of 2th elements is done by thread number 2
 addition of 4th elements is done by thread number 4
 Result array is 24 17 18 17 24
PS C:\Lab\HPC\2>
```

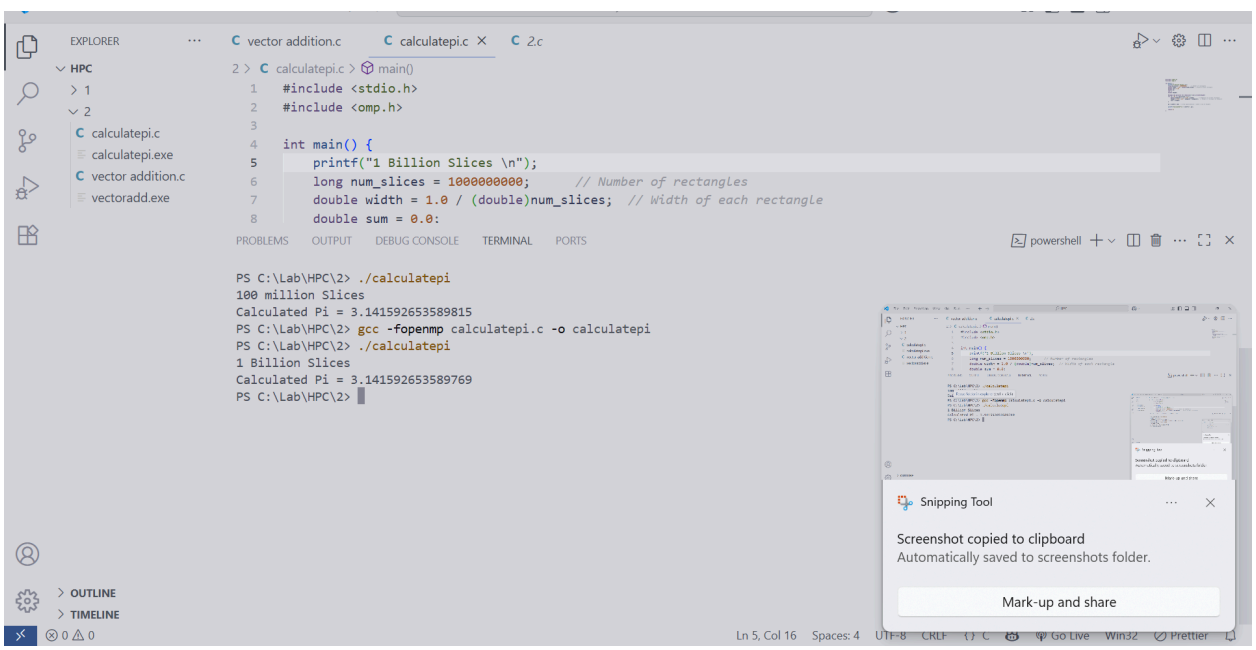
**Information:**
**Analysis:**

**Problem Statement 2:**
**Screenshots:**
**Code:**

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

int main() {
    printf("1 Billion Slices \n");
    long num_slices = 1000000000;       // Number of rectangles
    double width = 1.0 / (double)num_slices;  // Width of each rectangle
    double sum = 0.0;
```

```
    double pi;
    int i;
    double height;

    #pragma omp parallel for reduction(+:sum) private(height)
    for (i = 0; i < num_slices; i++) {
        double midpoint = (i + 0.5) * width;        // Midpoint of current
rectangle
        height = 4.0 / (1.0 + midpoint * midpoint);  // Height of
rectangle at midpoint
        sum += height;
    }

    pi = width * sum;  // Area approximation = width * sum of heights

    printf("Calculated Pi = %.15lf\n", pi);

    return 0;
}
```

**Result :**




**Information:**

**Analysis:**

22510021

Calculation of Pi using Midpoint Rule

Theoretical basis:

Known integral:

$$\pi = \int_0^1 \frac{4}{1+x^2}\,dx$$

the identity comes from inverse tangent function

$$\int \frac{1}{1+x^2}\,dx = \tan^{-1}(x)$$

① We divide the interval [0,1] into N slices (rectangle)

so width of each is $\text{width} = \frac{1}{N}$

② For each rectangle midpoint is

$$x_i = (i + 0.5) \times \text{width}$$

③ We Find height i.e. $y = f(x_i)$ for that point

$$f(x_i) = \frac{4}{1 + x_i^2}$$

Total area (approximation of $\pi$) is sum of all

$$\pi \approx \text{width} \times \sum_{i=0}^{N-1} \frac{4}{1 + ((i+0.5) \times \text{width})^2}$$

**Github Link:**

https://github.com/22510021-Shrikrishna/HPC.git