**Lab No 2: Date:**

**OpenMP programs on work-sharing constructs**

**Objectives:**

In this lab, student will be able to

 **Understand Work-Sharing Concepts:** Explain the concept of work sharing in OpenMP and identify situations where it is beneficial.

 **Implement Parallel Loops:** Write OpenMP directives to parallelize loops using the parallel for construct to distribute loop iterations across multiple threads.

**Lab Exercises:**

1. Write a OpenMP program to calculate 𝑝𝑜𝑤(𝑖, 𝑥) for all the threads where 𝑖 is an integer value and 𝑥 is the thread\_Id.
2. Write a OpenMP program that performs the sum of even numbers and odd numbers in a given input array. Create a separate thread to perform the sum of even numbers and odd numbers.
3. Write a OpenMP program to implement all the four basic operations of a calculator (Add, Sub, Mul, Div). Create a separate thread to perform the operations.
4. Write an OpenMP program to perform Arithmetic operations (Add, Sub, Mul, Div) on two vectors A and B of size 4.
5. Write a OpenMP program for generating prime numbers from a given starting number to the given ending number.
6. Write a program in OpenMP to toggle the character of a given character array indexed by the thread\_Id. Print the corresponding Thread\_Id.

Example: suppose the string is “HeLLo”, then the output should be “hEllO”.

1. Write a program using OpenMp to compute the Fibonacci number for the following arrays of numbers: A={10, 13, 5, 6}. Create a separate thread to perform the operations.
2. Write an OpenMP program to implement Matrix multiplication.
   1. Analyze the speedup and efficiency of the parallelized code.
   2. Vary the size of your matrices from 200, 400, 600, 800 and 1000 and measure the runtime with one thread and four threads.
   3. For each matrix size, change the number of threads from 2,4,6 and 8 and plot the speedup versus the number of threads. Compute the efficiency.
3. Write an OpenMP program to perform Matrix times vector multiplication. Vary the matrix and vector size and analyze the speedup and efficiency of the parallelized code.

**Additional programs**

1. Write an OpenMp program to read a matrix A of size 5x5. It produces a resultant matrix B of size 5x5. It sets all the principal diagonal elements of B matrix with 0. It replaces each row elements in the B matrix in the following manner. If the element is below the principal diagonal it replaces it with the maximum value of the row in the A matrix having the same row number of B. If the element is above the principal diagonal it replaces it with the minimum value of the row in the A matrix having the same row number of B. Analyze the speedup and efficiency of the parallelized code.
2. Write a parallel program using OpenMP that reads a matrix of size MxN and produce an output matrix B of same size such that it replaces all the non-border elements of A with its equivalent 1’s complement and remaining elements same as matrix A. Also produce a matrix D as shown below.

Example:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 6 | **2** | **7** | 3 |
| 2 | **3** | **5** | 1 |
| 9 | 1 | 2 | 5 |

A

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 6 | 5 | 8 | 3 |
| 2 | 4 | 10 | 1 |
| 9 | 1 | 2 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| 6 | **10** | **111** | 3 |
| 2 | **11** | **101** | 1 |
| 9 | 1 | 2 | 5 |

B D

1. Write a parallel program in OpenMP to reverse the digits of the following integer array of size 9. Initialize the input array to the following values:
   1. Input array: 18, 523, 301, 1234, 2, 14, 108, 150, 1928
   2. Output array: 81, 325, 103, 4321, 2, 41, 801, 51, 8291

**Steps to Analyze Speedup and Efficiency**

**Compile the Program**:

**Run the Program**:

**Record the Execution Times**:

**Calculate Speedup**: Speedup S is calculated as:

where T1​ is the execution time with one thread and Tp​ is the execution time with p threads.

**Calculate Efficiency**: Efficiency E is calculated as:

