Dated :
Assessment No. : 3

SCENARIO – I

Write a simple OpenMP program to employ a *'reduction'* clause to express the reduction of a for loop. In order to specify the reduction in OpenMP, we must provide

1. An operation (+ / * / o)
2. A reduction variable (sum / product / reduction). This variable holds the result of the computation.

Example:

Let's parallelize the following for loop:

```
sum = 0;
#pragma omp parallel for shared(sum, a) reduction(+: sum)
for (auto i = 0; i < 9; i++)
{
   sum += a[i]
}
```

Each thread has sumloc, which is a local copy of the reduction variable. The threads then perform the following computations

- **Thread 1**

sumloc_1 = a[0] + a[1] + a[2]

- **Thread 2**

sumloc_2 = a[3] + a[4] + a[5]

- **Thread 3**

sumloc_3 = a[6] + a[7] + a[8]

In the end, when the treads join together, OpenMP reduces local copies to the shared reduction variable

sum = sumloc_1 + sumloc_2 + sumloc_3

1. Show the complete code execution of above logic using C/C++

BRIEF ABOUT YOUR APPROACH:

Dated :
Assessment No. : 3

SOURCE CODE:

EXECUTION:

RESULTS:

Dated :
Assessment No. : 3

SCENARIO – II

Write an OpenMP program to find the smallest element in a list of numbers using OpenMP REDUCTION clause.

Description

Largest element in a list of numbers is found using OpenMP PARALLEL DO directive and REDUCTION clause. Reductions are a sufficiently common type of operation. OpenMP includes a reduction data scope clause just to handle the variable. In reduction, we repeatedly apply a binary operator to a variable and some other value, and store the result back in the variable. In this example we have added the clause REDUCTION (MAX : LargeNumber), which tells the compiler that LargeNumber is the target of a sum reduction operation.

BRIEF ABOUT YOUR APPROACH:

SOURCE CODE:

EXECUTION:

RESULTS:

Dated :
Assessment No. : 3

SCENARIO – III

Write an OpenMP program to find the Max and Min elements in a list of numbers using OpenMP Critical clause to understand:

Hint: As Max & Min value is easily prone to change by another thread after comparing with Array [Index], the use of *'critical'* section is highly demanded to execute such computation on one thread at a time.

Example:

```
#pragma omp critical
        {
          if (Array Index is greater/lesser than Max/Min)
        }
```

BRIEF ABOUT YOUR APPROACH:

SOURCE CODE:

EXECUTION:

RESULTS: