**Assignment 03:**

**To develop a distributed system, to find the sum of N elements in an array by distributing N/n elements to n number of processors MPI or OpenMP. Demonstrate by displaying the intermediate sums calculated at different processors.**

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

int main(int argc, char \*\*argv) { int rank, size; int N = 10; // total number of elements int n = 4; // number of processors int\* arr = malloc(sizeof(int) \* N); // allocate memory for the array int i, local\_sum = 0, global\_sum = 0;

// initialize the array with sequential values for (i = 0; i < N; i++) { arr[i] = i + 1;

}

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank); MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

if (size != n) {

printf("Error: must run with %d processes\n", n); MPI\_Finalize(); return 1;

}

// calculate the local sum int start = rank \* N / size; int end = (rank + 1) \* N / size; for (i = start; i < end; i++) { local\_sum += arr[i];

}

// reduce the local sums to get the global sum

MPI\_Reduce(&local\_sum, &global\_sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

// send the local sum to process 0 if (rank != 0) {

MPI\_Send(&local\_sum, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

} else {

// process 0 receives the local sums and prints the intermediate and final results printf("Rank %d local sum: %d\n", rank, local\_sum); for (i = 1; i < size; i++) {

MPI\_Recv(&local\_sum, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD,

MPI\_STATUS\_IGNORE);

printf("Rank %d local sum: %d\n", i, local\_sum);

}

printf("Global sum: %d\n", global\_sum); fflush(stdout);

}

MPI\_Finalize();

free(arr);

return 0;

}

Output:

