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: