**Write a MPI Program to demonstration of MPI_Reduce and MPI_Allreduce (MPI_MAX, MPI_MIN, MPI_SUM, MPI_PROD)**

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char** argv) {
   MPI_Init(&argc, &argv);  // Initialize MPI
   int rank, size;
   int value, sum, prod, max, min;
   int all_sum, all_prod, all_max, all_min;
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);  // Get this process's ID
   MPI_Comm_size(MPI_COMM_WORLD, &size);  // Get total number of processes
   value = rank + 1;  // Each process sets its value to (rank + 1)
   // --- MPI_Reduce: results are available only on root (rank 0) ---
   MPI_Reduce(&value, &sum, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
   MPI_Reduce(&value, &prod, 1, MPI_INT, MPI_PROD, 0, MPI_COMM_WORLD);
   MPI_Reduce(&value, &max, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);
   MPI_Reduce(&value, &min, 1, MPI_INT, MPI_MIN, 0, MPI_COMM_WORLD);
   if (rank == 0) {
      printf("== Results using MPI_Reduce (only at root) ==\n");
      printf("Sum  = %d\n", sum);
      printf("Prod = %d\n", prod);
      printf("Max  = %d\n", max);
      printf("Min  = %d\n", min);
   }
   // --- MPI_Allreduce: results are available to ALL processes ---
   MPI_Allreduce(&value, &all_sum, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
   MPI_Allreduce(&value, &all_prod, 1, MPI_INT, MPI_PROD, MPI_COMM_WORLD);
   MPI_Allreduce(&value, &all_max, 1, MPI_INT, MPI_MAX, MPI_COMM_WORLD);
   MPI_Allreduce(&value, &all_min, 1, MPI_INT, MPI_MIN, MPI_COMM_WORLD);
   printf("Process %d - Allreduce: Sum=%d, Prod=%d, Max=%d, Min=%d\n",
         rank, all_sum, all_prod, all_max, all_min);
```

```
    MPI_Finalize();  // Finalize MPI
    return 0;
}
```

**Output:**

```
mpicc reduce_allreduce_demo.c -o reduce_allreduce_demo
mpirun ./reduce_allreduce_demo
== Results using MPI_Reduce (only at root) ==
Sum  = 3
Prod = 2
Max  = 2
Min  = 1
Process 0 - Allreduce: Sum=3, Prod=2, Max=2, Min=1
Process 1 - Allreduce: Sum=3, Prod=2, Max=2, Min=1
```