5. Write a MPI Program to demonstration of MPI_Send and MPI_Recv.

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char** argv) {
   MPI_Init(&argc, &argv);  // Initialize MPI
   int rank, size, data;
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);  // Get current process ID
   MPI_Comm_size(MPI_COMM_WORLD, &size);  // Get total number of processes
   if (size < 2) {
      if (rank == 0)
         printf("This program requires at least 2 processes.\n");
      MPI_Finalize();
      return 0;
   }
   if (rank == 0) {
      data = 42;
      MPI_Send(&data, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
      printf("Process 0 sent data: %d to Process 1\n", data);
   }
   if (rank == 1) {
      MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      printf("Process 1 received data: %d from Process 0\n", data);
   }
   MPI_Finalize();  // Finalize MPI
   return 0;
}
```

Output:

```
mpicc send_recv_simple.c -o send_recv_simple
mpi_programs$ mpirun ./send_recv_simple
Process 1 received data: 42 from Process 0
Process 0 sent data: 42 to Process 1
```