```c
/* Joshua Catoe
   CSCI 473
   Exam Question 4
*/

#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

int main() {

    MPI_Init(NULL,NULL);

    int size;             // Number of processes
    int my_rank;          // Rank of current process
    int r_rank;           // Rank of right neighbor
    int l_rank;           // Rank of left neighbor
    int my_data;          // Current process's data to send
    int r_buf;            // Data received from right neighbor
    int l_buf;            // Data received from left neighbor

    MPI_Comm_size(MPI_COMM_WORLD, &size);  // Get number of processes

    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);  // Get rank of process

    // Determine left and right neighbors (0 and size-1 "wrap around")
    if(my_rank==0) {

        r_rank=my_rank+1;
        l_rank=size-1;
    }
    else if(my_rank==size-1) {

        r_rank=0;
        l_rank=my_rank-1;
    }
    else {

        r_rank=my_rank+1;
        l_rank=my_rank-1;
    }

    srandom(my_rank);      // Seed random() with rank
    my_data=random()%100;  // Modulo shortens the range of random()

    printf("Process [ %i] has myData = %i, R_rank = %i, L_rank = %i\n",my_rank,my_data,r_rank,l_rank);

    // Send to right, receive from left
    MPI_Sendrecv(&my_data,1,MPI_INT,r_rank,19,&l_buf,1,MPI_INT,l_rank,19,MPI_COMM_WORLD,MPI_STATUS_IGNORE);

    // Send to left, receive from right
    MPI_Sendrecv(&my_data,1,MPI_INT,l_rank,19,&r_buf,1,MPI_INT,r_rank,19,MPI_COMM_WORLD,MPI_STATUS_IGNORE);

    printf("Process [ %i]: recv'd %i from R_rank = %i, recv'd %i from L_rank = %i\n",my_rank,r_buf,r_rank,l_buf,l_rank);

    MPI_Finalize();

    return 0;
}
```