```
/********************************************************************
 * pingpong with "big data structure" :) betweem two processes
 *
 ********************************************************************/

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

#define MAXSIZE 1000000


int main(int argc, char *argv[]) {
  int rank, size;
  double a, b;
  int dest, source, rc, count;
  int* bigdata = new int[MAXSIZE];
  MPI_Status status;

  MPI_Init(&argc, &argv);                /* Initialize MPI               */
  MPI_Comm_size(MPI_COMM_WORLD, &size); /* Get the number of processors */
  MPI_Comm_rank(MPI_COMM_WORLD, &rank); /* Get my number                */

  // test variables: a and bigdata
  a = 100.0 + (double) rank;  /* Different a on different processors */

  for (int i=0; i<MAXSIZE;i++)
        bigdata[i] = i;

  /* Exchange variable a, notice the send-recv order */
  /* Change Send-Recv order to test MPI blocking modes! */
  // simple double NO DEADLOCK, big vector YES DEADLOCK (change Send/RECV order!)
  if (rank == 0) {
    dest = 1;
    source = 1;
    MPI_Send(&bigdata[0], MAXSIZE, MPI_INT, dest, 17, MPI_COMM_WORLD);
    MPI_Recv(bigdata, MAXSIZE, MPI_INT, source, 23, MPI_COMM_WORLD, &status);
    //MPI_Send(&a, 1, MPI_DOUBLE, dest, 17, MPI_COMM_WORLD);
    //MPI_Recv(&b, 1, MPI_DOUBLE, source, 23, MPI_COMM_WORLD, &status);
    printf("Processor 0 got %f from processor 1\n", b);
  } else if (rank==1) {
    dest = 0;
    source = 0;
    //MPI_Send(&a, 1, MPI_DOUBLE, source, 23, MPI_COMM_WORLD);
    //MPI_Recv(&b, 1, MPI_DOUBLE, dest, 17, MPI_COMM_WORLD, &status);
    MPI_Recv(bigdata, MAXSIZE, MPI_INT, dest, 17, MPI_COMM_WORLD, &status);
    MPI_Send(bigdata, MAXSIZE, MPI_INT, dest, 23, MPI_COMM_WORLD);

    printf("Processor 1 got %f from processor 0\n", b);
  }

  MPI_Get_count(&status, MPI_DOUBLE, &count);  // how many doubles?
  //MPI_Get_count(&status, MPI_CHAR, &count); // how many bytes? (or MPI_CHAR_BYTE)
  printf("Task %d : Received %d doubles from task %d with tag %d \n", rank, count, status.MPI_

  delete[] bigdata;

  MPI_Finalize();

  return 0;
}
```