```
#include <stdio.h>
#include <string.h>
#include <mpi.h>

using std::cout;
using std::endl;

const int MAX_STRING = 100;

int main(int argc, char **argv) {
    char    greeting[MAX_STRING];
    int     comm_sz;    /* number of processes  */
    int     my_rank;    /* process rank         */

    //Initialize the command line arguments on every process
    MPI_Init(&argc, &argv);

    int token = 0;
    token = atoi(argv[1]);

    //Get the number of processes in MPI_COMM_WORLD, and put
    //it in the 'comm_sz" variable; ie., how many processes
    //are running this program (the same as what you put in the
    //-np argument).
    MPI_Comm_size(MPI_COMM_WORLD, &comm_sz);

    //Get the rank of this particular process in MPI_COMM_WORLD,
    //and put it in the 'my_rank' variable -- ie., what number
    //is this process
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    int next_rank = my_rank + 1;
    if (next_rank >= comm_sz) next_rank = 0;

    int prev_rank = my_rank - 1;
    if (prev_rank < 0) prev_rank = comm_sz - 1;

    /**
     *  Status/Requests so we can wait on the asynchronous sends/receives
     */
    MPI_Request request;
    MPI_Status status;

    /**
     * The head process starts the ring by sending the first token
     */
    if (my_rank == 0) {
        if (token > 0) token--;

        cout << "[process " << my_rank << "]: sending  " << token << endl;

        MPI_Isend(&token,
                  1,
                  MPI_INT,
                  next_rank,
                  0,
                  MPI_COMM_WORLD,
                  &request);

        MPI_Wait(&request, &status);
    }

    /**
     *  Each process will receive the token from the previous in the ring, and sending
```

```
       *  it along until the token reaches 0.
       */
      while (token - comm_sz >= 0) {
          MPI_Irecv(&token,
                    1,
                    MPI_INT,
                    prev_rank,
                    0,
                    MPI_COMM_WORLD,
                    &request);

          //You could do some processing in here if you wanted.

          MPI_Wait(&request, &status);

  //        cout << "[process " << my_rank << "]: received " << token << endl;
          token--;

          if (token >= 0) {
              cout << "[process " << my_rank << "]: sending  " << token << endl;
              MPI_Isend(&token,
                        1,
                        MPI_INT,
                        next_rank,
                        0,
                        MPI_COMM_WORLD,
                        &request);

              MPI_Wait(&request, &status);
          }
      }

      /*Inform MPI that this process has finished*/
      MPI_Finalize();
      return 0;
  }
```