```cpp
#include <iostream>
#include <stdio.h>
#include <string.h>
#include "mpi.h" // message passing interface
using namespace std;

//
// Program 1
// Two Rings - 10 points
//  Rachel Burke
//

int main (int argc, char * argv[]) {

    int my_rank;            // my CPU number for this process
    int p;                  // number of CPUs that we have
    int source;             // rank of the sender
    int dest;               // rank of destination
    int tag = 0;            // message number
    char message[100];      // message itself
    MPI_Status status;      // return status for receive

    // Start MPI
    MPI_Init(&argc, &argv);

    // Find out my rank!
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Find out the number of processes!
    MPI_Comm_size(MPI_COMM_WORLD, &p);

    // The Real Program Starts Here!
    char *baton = message;
    sprintf(message, "Sent list: ");

    // Even Ring Start
    if (my_rank == 0)
    {
        cout << "Baton at: " << my_rank << endl;
        sprintf(baton + strlen(baton), "%d ", my_rank);
        MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, 2, tag, MPI_COMM_WORLD);
        cout << "Baton sent to: 2" << endl;
        MPI_Recv(baton, 100, MPI_CHAR, p - ((p + 1) % 2) - 1, tag, MPI_COMM_WORLD,
        &status);
        cout << "Baton back at: " << my_rank << "\n" << baton << endl;
    } // end if

    // Odd Ring Start
    else if (my_rank == 1)
    {
        cout << "Baton at: " << my_rank << endl;
        sprintf(baton + strlen(baton), "%d ", my_rank);
        MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, p - (p % 2) - 1, tag,
        MPI_COMM_WORLD);
        cout << "Baton sent to: " << p - (p % 2) - 1 << endl;
        MPI_Recv(baton, 100, MPI_CHAR, 3, tag, MPI_COMM_WORLD, &status);
        cout << "Baton back at: " << my_rank << "\n" << baton << endl;
    } // end if

    // Even Number of Processors
    else if(p % 2 == 0)
    {
        // Even Ring
        if (my_rank % 2 == 0)
        {
            MPI_Recv(baton, 100, MPI_CHAR, my_rank - 2, tag, MPI_COMM_WORLD, &status);
            cout << "Baton at: " << my_rank << "\nBaton sent to: " << (my_rank + 2) % p
            << endl;
            sprintf(baton + strlen(baton), "%d ", my_rank);
```

```
67              MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, (my_rank + 2) % p, tag,
                MPI_COMM_WORLD);
68          } // end if
69          // Odd Ring
70          else
71          {
72              MPI_Recv(baton, 100, MPI_CHAR, (my_rank + 2) % p, tag, MPI_COMM_WORLD,
                &status);
73              cout << "Baton at: " << my_rank << "\nBaton sent to: " << my_rank - 2 <<endl;
74              sprintf(baton + strlen(baton), "%d ", my_rank);
75              MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, my_rank - 2, tag,
                MPI_COMM_WORLD);
76          } //end else
77      }// end else if
78
79      // Odd Number of Processors
80      else
81      {
82          // Even Ring
83          if (my_rank % 2 == 0)
84          {
85              MPI_Recv(baton, 100, MPI_CHAR, my_rank - 2, tag, MPI_COMM_WORLD, &status);
86              cout << "Baton at: " << my_rank << "\nBaton sent to: " << (my_rank + 2) %
                (p + 1) << endl;
87              sprintf(baton + strlen(baton), "%d ", my_rank);
88              MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, (my_rank + 2) % (p + 1), tag,
                MPI_COMM_WORLD);
89          } // end if
90          // Odd Ring
91          else
92          {
93              MPI_Recv(baton, 100, MPI_CHAR, (my_rank + 2) % (p - 1), tag,
                MPI_COMM_WORLD, &status);
94              cout << "Baton at: " << my_rank << "\nBaton sent to: " << my_rank - 2 <<endl;
95              sprintf(baton + strlen(baton), "%d ", my_rank);
96              MPI_Send(baton, strlen(baton) + 1, MPI_CHAR, my_rank - 2, tag,
                MPI_COMM_WORLD);
97          } //end else
98      }
99
100     // Shut down MPI
101     MPI_Finalize();
102
103     return 0;
104 }
```