

```

/* Odd-Even Sort */

#include <stdlib.h>
#include <mpi.h>      /* Include MPI's header file */

int IncOrder(const void *el, const void *e2);
void CompareSplit (int nlocal, int *elmnts, int *relmnts,
                  int *wspace, int keepsmall);

main(int argc, char *argv[])
{
    int n;           /* The total number of elements to be sorted */
    int npes;       /* The total number of processes */
    int myrank;     /* The rank of the calling process */
    int nlocal;     /* The local number of elements */
    int *elmnts;    /* The array that stores the local elements */
    int *relmnts;   /* The array that stores the received elements */
    int oddrank;    /* The rank of the process during odd-phase communication */
    int evenrank;   /* The rank of the process during even-phase communication */
    int *wspace;    /* Working space during the compare-split operation */
    int i;
    MPI_Status status;

    /* Initialize MPI and get system information */
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &npes);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

    n = atoi(argv[1]);
    nlocal = n/npes; /* Compute the number of elements to be stored locally */

    /* Allocate memory for the various arrays */
    elmnts = (int *)malloc(nlocal*sizeof(int));
    relmnts = (int *)malloc(nlocal*sizeof(int));
    wspace = (int *)malloc(nlocal*sizeof(int));

    /* Fill-in elmnts array with random elements */
    srandom(myrank);
    for (i=0; i<nlocal; i++)
        elmnts [i] = random();

    /* Sort the local elements using the built-in quicksort routine */
    qsort(elmnts, nlocal, sizeof(int), IncOrder);

    /* Determine the rank of the processors that myrank needs to communicate */
    /* during the odd and even phases of the algorithm */

    if (myrank % 2 == 0) {
        oddrank = myrank-1;
        evenrank = myrank+1;
    } else {
        oddrank = myrank+1;
        evenrank = myrank-1;
    }

    /* disable ring communication */
    if (oddrank == -1 || oddrank == npes)
        oddrank = MPI_PROC_NULL;
    if (evenrank == -1 || evenrank == npes)
        evenrank = MPI_PROC_NULL;

    /* Get into the main loop of the odd-even sorting algorithm */
    for (i=0; i<npes-1; i++) {
        if (i%2 == 1) /* Odd phase*/
            MPI_Sendrecv(elmnts, nlocal, MPI_INT, oddrank, 1,
                          relmnts, nlocal, MPI_INT, oddrank, 1,
                          MPI_COMM_WORLD, &status);

        else          /* Even phase */

```

```

    MPI_Sendrecv(elmnts, nlocal, MPI_INT, evenrank, 1,
                 relmnts, nlocal, MPI_INT, evenrank, 1,
                 MPI_COMM_WORLD, &status);

    CompareSplit(nlocal, elmnts, relmnts, wspace, myrank < status.MPI_SOURCE);
}

free(elmnts);
free(relmnts);
free(wspace);

MPI_Finalize();
}

/* This is the CompareSplit Function */
void CompareSplit (int nlocal, int *elmnts, int *relmnts,
                  int *wspace, int keepsmall)
{
    int i, j, k;
    for (i=0; i<nlocal; i++)
        wspace [i] = elmnts [i] ;    /* Copy elmnts array into the wspace array */

    if (keepsmall) { /* Keep the nlocal smaller elements */
        for (i=j=k=0; k<nlocal; k++) {
            if (j==nlocal || (i<nlocal && wspace[i] < relmnts[j]))
                elmnts[k] = wspace[i++];
            else
                elmnts[k] = relmnts[j++];
        }
    }
    else { /* Keep the nlocal larger elements */
        for (i=k=nlocal-1, j=nlocal-1; k>=0; k--) {
            if (j==0 || (i>=0 && wspace[i] >= relmnts[j]))
                elmnts[k] = wspace[i--];
            else
                elmnts[k] = relmnts[j--];
        }
    }
}

/* The IncOrder function that is called by qsort is defined as follows */
int IncOrder(const void *e1, const void *e2)
{
    return (*((int*)e1) - *((int*)e2));
}

```