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

int main(int argc, char* argv[]) {
    MPI_Init(&argc, &argv);
    // Get number of processes and check that 4 processes are used
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if(size != 4) {
        printf("This application is meant to be run with 4 MPI processes.\n");
        MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
    }

    // Determine root's rank
    int root_rank = 0;
    // Get my rank
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Define my value
    int my_value[2];
    my_value[0] = my_rank * 100;
    my_value[1] = my_rank;
    printf("Process %d, my value = %d, %d.\n", my_rank, my_value[0], my_value[1]);

    if(my_rank == root_rank) {
        int buffer[8];
        MPI_Gather(&my_value, 2, MPI_INT, buffer, 2, MPI_INT, root_rank, MPI_COMM_WORLD);
        printf("Values collected on process %d: %d, %d, %d, %d, %d, %d, %d, %d.\n", 
my_rank, buffer[0], buffer[1], buffer[2], buffer[3], buffer[4], buffer[5], buffer[6], 
buffer[7]);
    } else {
        MPI_Gather(&my_value, 2, MPI_INT, NULL, 0, MPI_INT, root_rank, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

```
> mpicc gather.c
> mpirun -n 4 ./a.out
Process 0, my value = 0, 0.
Process 2, my value = 200, 2.
Process 3, my value = 300, 3.
Process 1, my value = 100, 1.
Values collected on process 0: 0, 0, 100, 1, 200, 2, 300, 3.
```

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

int main(int argc, char* argv[]) {
    MPI_Init(&argc, &argv);
    // Get number of processes and check that 4 processes are used
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if(size != 4) {
        printf("This application is meant to be run with 4 MPI processes.\n");
        MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
    }
    // Determine root's rank
    int root_rank = 0;
    // Get my rank
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Define my value
    int my_value[2];

    if(my_rank == root_rank) {
        int buffer[] = {0, 1, 2, 3, 4, 5, 6, 7};
        printf("Values to scatter from process %d: %d, %d, %d, %d, %d, %d, %d, %d\n",
my_rank, buffer[0], buffer[1], buffer[2], buffer[3], buffer[4], buffer[5], buffer[6],
buffer[7]);
        MPI_Scatter(buffer, 2, MPI_INT, &my_value, 2, MPI_INT, root_rank,
MPI_COMM_WORLD);
    } else {
        MPI_Scatter(NULL, 2, MPI_INT, &my_value, 2, MPI_INT, root_rank, MPI_COMM_WORLD);
    }

    printf("Process %d, my value = {%d, %d}\n", my_rank, my_value[0], my_value[1]);

    MPI_Finalize();
    return 0;
}
```

```
> mpicc scatter.c
> mpirun -n 4 ./a.out
Values to scatter from process 0: 0, 1, 2, 3, 4, 5, 6, 7
Process 0, my value = {0, 1}
Process 1, my value = {2, 3}
Process 2, my value = {4, 5}
Process 3, my value = {6, 7}
```

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

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

    // Get number of processes and check that 4 processes are used
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if (size != 4) {
        printf("This application is meant to be run with 4 processes.\n");
        MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
    }

    // Determine root's rank
    int root_rank = 0;

    // Get my rank
    int my_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

    // Define my value
    int my_value[2];

    if (my_rank == root_rank) {
        int buffer[20] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
18, 19};
```

```
        printf("Values to scatter from process %d: %d, %d, %d, %d, %d, %d, %d, %d, %d,
%d, %d, %d, %d, %d, %d, %d, %d, %d, %d, %d \n", my_rank, buffer[0], buffer[1], buffer[2],
buffer[3], buffer[4], buffer[5], buffer[6], buffer[7], buffer[8], buffer[9], buffer[10],
buffer[11], buffer[12], buffer[13], buffer[14], buffer[15], buffer[16], buffer[17],
buffer[18], buffer[19]);
        MPI_Scatter(buffer, 5, MPI_INT, &my_value, 5, MPI_INT, 0, MPI_COMM_WORLD);
    } else {
        MPI_Scatter(NULL, 5, MPI_INT, &my_value, 5, MPI_INT, 0, MPI_COMM_WORLD);
    }

    printf("Process %d, my value = (%d, %d, %d, %d, %d).\n", my_rank, my_value[0],
my_value[1], my_value[2], my_value[3], my_value[4]);
    int sum = my_value[0] + my_value[1] + my_value[2] + my_value[3] + my_value[4];

    if (my_rank == root_rank) {
        int buffer[4];
        MPI_Gather(&sum, 1, MPI_INT, buffer, 1, MPI_INT, root_rank, MPI_COMM_WORLD);
        printf("Values collected on process %d: %d, %d, %d, %d.\n", my_rank, buffer[0],
buffer[1], buffer[2], buffer[3]);
    } else {
        MPI_Gather(&sum, 1, MPI_INT, NULL, 0, MPI_INT, root_rank, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return EXIT_SUCCESS;
}

> mpicc scatter_gather.c
> mpirun -n 4 ./a.out
Values to scatter from process 0: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
16, 17, 18, 19
Process 2, my value = (10, 11, 12, 13, 14).
Process 0, my value = (0, 1, 2, 3, 4).
Process 1, my value = (5, 6, 7, 8, 9).
Process 3, my value = (15, 16, 17, 18, 19).
Values collected on process 0: 10, 35, 60, 85.
```