Slot 1

```
#include
#include
#include

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

int rank, size;
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

// Calculate the number of elements each process will handle
int elements_per_process = 1000 / size;

// Initialize a seed for random number generation based on the rank
srand(rank);

// Generate random numbers on each process
int local_numbers[elements_per_process];
for (int i = 0; i < elements_per_process; i++) {
local_numbers[i] = rand();
}

// Gather all local numbers to process 0
int all_numbers[1000];
MPI_Gather(local_numbers, elements_per_process, MPI_INT, all_numbers, elements_per_process,
MPI_INT, 0, MPI_COMM_WORLD);

// Print the numbers on process 0
if (rank == 0) {
printf("Generated Numbers:\n");
for (int i = 0; i < 1000; i++) {
printf("%d ", all_numbers[i]);
}
printf("\n");
}

MPI_Finalize();
return 0;
}
```

Slot 2

```
#include
#include
#include

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

MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

int *randomNumbers = (int *)malloc(1000 * sizeof(int));
for (int i = 0; i < 1000; i++) {
randomNumbers[i] = rand() % 1000; // Generating random numbers between 0 and 999
}

int localMin = randomNumbers[0];
int localMax = randomNumbers[0];
```

```
for (int i = 1; i < 1000; i++) {
if (randomNumbers[i] < localMin) {
localMin = randomNumbers[i];
}
if (randomNumbers[i] > localMax) {
localMax = randomNumbers[i];
}
}

int globalMin, globalMax;

MPI_Reduce(&localMin, &globalMin, 1, MPI_INT, MPI_MIN, 0, MPI_COMM_WORLD);
MPI_Reduce(&localMax, &globalMax, 1, MPI_INT, MPI_MAX, 0, MPI_COMM_WORLD);

if (rank == 0) {
printf("Minimum number: %d\n", globalMin);
printf("Maximum number: %d\n", globalMax);
}

free(randomNumbers);
MPI_Finalize();

return 0;
}
```