// MPI Program to Calculate Pi using MPI\_Bcast and MPI\_Reduce

#include <mpi.h>

#include <stdio.h>

#include <math.h>

#define NUM\_STEPS 100000

double compute\_pi(int start, int end, double step) {

double sum = 0.0;

for (int i = start; i < end; i++) {

double x = (i + 0.5) \* step;

sum += 4.0 / (1.0 + x \* x);

}

return sum;

}

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

int rank, size;

double step = 1.0 / (double)NUM\_STEPS;

double local\_sum = 0.0, global\_sum = 0.0;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

int chunk\_size = NUM\_STEPS / size;

int start = rank \* chunk\_size;

int end = (rank == size - 1) ? NUM\_STEPS : start + chunk\_size;

local\_sum = compute\_pi(start, end, step);

MPI\_Reduce(&local\_sum, &global\_sum, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0) {

double pi = step \* global\_sum;

printf("Approximate value of Pi: %lf\n", pi);

}

MPI\_Finalize();

return 0;

}

// MPI Program to Find Prime Numbers using MPI\_Send and MPI\_Recv

#include <mpi.h>

#include <stdio.h>

#include <stdbool.h>

bool is\_prime(int n) {

if (n < 2) return false;

for (int i = 2; i \* i <= n; i++) {

if (n % i == 0) return false;

}

return true;

}

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

int rank, size, num;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

int max\_value = 100;

if (rank == 0) {

for (int i = 2; i <= max\_value; i++) {

int worker;

MPI\_Recv(&worker, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Send(&i, 1, MPI\_INT, worker, 0, MPI\_COMM\_WORLD);

}

for (int i = 1; i < size; i++) {

num = -1;

MPI\_Send(&num, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

}

} else {

while (1) {

MPI\_Send(&rank, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

MPI\_Recv(&num, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

if (num < 0) break;

if (is\_prime(num)) {

printf("Prime: %d\n", num);

}

}

}

MPI\_Finalize();

return 0;

}