%%writefile omp\_statss.cpp

#include <iostream>

#include <omp.h>

#include <climits> // for INT\_MAX and INT\_MIN

using namespace std;

void min(int \*arr, int n) {

int min\_value = INT\_MAX;

#pragma omp parallel for reduction(min : min\_value)

for (int i = 0; i < n; i++) {

if (arr[i] < min\_value) {

min\_value = arr[i];

}

}

cout << "\nMin value: " << min\_value << endl;

}

void max(int \*arr, int n) {

int max\_value = INT\_MIN;

#pragma omp parallel for reduction(max : max\_value)

for (int i = 0; i < n; i++) {

if (arr[i] > max\_value) {

max\_value = arr[i];

}

}

cout << "Max value: " << max\_value << endl;

}

void avg(int \*arr, int n) {

int sum = 0;

#pragma omp parallel for reduction(+:sum)

for (int i = 0; i < n; i++) {

sum += arr[i];

}

float average = (float)sum / n;

cout << "Sum: " << sum << endl;

cout << "Average: " << average << endl;

}

int main() {

omp\_set\_num\_threads(4);

int n;

cout << "Enter the number of elements in the array: ";

cin >> n;

int \*arr = new int[n];

cout << "Enter the elements of the array: ";

for (int i = 0; i < n; ++i) {

cin >> arr[i];

}

cout << "\nArray elements are: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

min(arr, n);

max(arr, n);

avg(arr, n);

delete[] arr;

return 0;

}