Max.c

#include <stdio.h>

#include <omp.h>

int main() {

double arr[10];

omp\_set\_num\_threads(4);

double max\_val = 0.0;

int i;

// Initialize the array

for (i = 0; i < 10; i++)

arr[i] = 2.0 + i;

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

for (i = 0; i < 10; i++) {

printf("thread id = %d and i = %d \n", omp\_get\_thread\_num(), i);

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

max\_val = arr[i];

}

}

printf("\nmax\_val = %f", max\_val);

return 0;

}

Min.c

#include <stdio.h>

#include <omp.h>

int main() {

double arr[10];

omp\_set\_num\_threads(4);

double min\_val = 9.0; // Initial value set to a large number

int i;

// Initialize the array

for (i = 0; i < 10; i++)

arr[i] = 2.0 + i;

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

for (i = 0; i < 10; i++) {

printf("thread id = %d and i = %d \n", omp\_get\_thread\_num(), i);

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

min\_val = arr[i];

}

}

printf("\nmin\_val = %f", min\_val);

return 0;

}

Sum.c#include <stdio.h>

#include <omp.h>

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

int i, n;

float a[100], b[100], sum;

/\* Some initializations \*/

n = 3;

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

a[i] = b[i] = i \* 1.0;

sum = 0.0;

#pragma omp parallel for reduction(+:sum)

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

sum += (a[i] \* b[i]);

printf("Sum = %f\n", sum);

return 0;

}

Avg.c

#include <iostream>

#include <omp.h>

using namespace std;

int main() {

int a[100], n, i;

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

cin >> n;

cout << "\nEnter array elements: ";

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

cin >> a[i];

}

cout << "\nArray elements are:\t";

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

cout << a[i] << "\t";

}

float avg = 0, sum = 0;

#pragma omp parallel

{

int id = omp\_get\_thread\_num();

#pragma omp for reduction(+:sum)

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

sum += a[i];

cout << "\nFor i = " << i << " thread " << id << " is executing" << endl;

}

}

avg = sum / n;

cout << "Output = " << avg << endl;

return 0;

}