**Code :**

#include <iostream>

#include <omp.h>

#include <time.h>

#include <iomanip>

#include <limits>

using namespace std;

int sum(int a[],int n) {

int sum = 0;

#pragma omp parallel for reduction(+:sum)

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

sum += a[i];

}

return sum;

}

int min(int a[],int n) {

int v = a[0];

#pragma omp parallel for reduction(min:v)

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

if(a[i] < v)

v = a[i];

}

return v;

}

int max(int a[],int n) {

int v = a[0];

#pragma omp parallel for reduction(max:v)

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

if(a[i] > v)

v = a[i];

}

return v;

}

float avg(int a[],int n) {

return sum(a,n)/n;

}

int main()

{

int a[100];

int cnt=0;

for(int i = 0;i<100;i++)

{ a[i] = i+5;

cnt = cnt +1;}

cout<<" Input data is :" ;

for(int i = 0;i<cnt;i++)

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

cout<<"\n";

struct timespec start, end;

// start timer.

// clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, &start);

//lo clock\_gettime(CLOCK\_REALTIME, &start);

clock\_gettime(CLOCK\_MONOTONIC, &start);

// unsync the I/O of C and C++.

ios\_base::sync\_with\_stdio(false);

cout<<"Sum : "<<sum(a,100);

cout<<"\n";

// stop timer.

// clock\_gettime(CLOCK\_PROCESS\_CPUTIME\_ID, &end);

// clock\_gettime(CLOCK\_REALTIME, &end);

clock\_gettime(CLOCK\_MONOTONIC, &end);

// Calculating total time taken by the program.

double time\_taken;

time\_taken = (end.tv\_sec - start.tv\_sec) \* 1e9;

time\_taken = (time\_taken + (end.tv\_nsec - start.tv\_nsec)) \* 1e-9;

cout << "Time taken by program is : " << fixed

<< time\_taken << setprecision(5);

cout << " sec" << endl;

cout<<"\n";

cout<<" Sum of 100 numbers: "<<sum(a,100);

cout<<"\n";

cout<<" Minimum number from 100 numbers: "<<min(a,100);

cout<<"\n";

cout<<" Maximum number from 100 numbers: "<<max(a,100);

cout<<"\n";

cout<<" Average of 100 numbers: "<<avg(a,100);

cout<<"\n";

return 0;

}

**Output:**

