Experiment No: 3. Implement Min, Max, Sum and Average operations using Parallel Reduction.

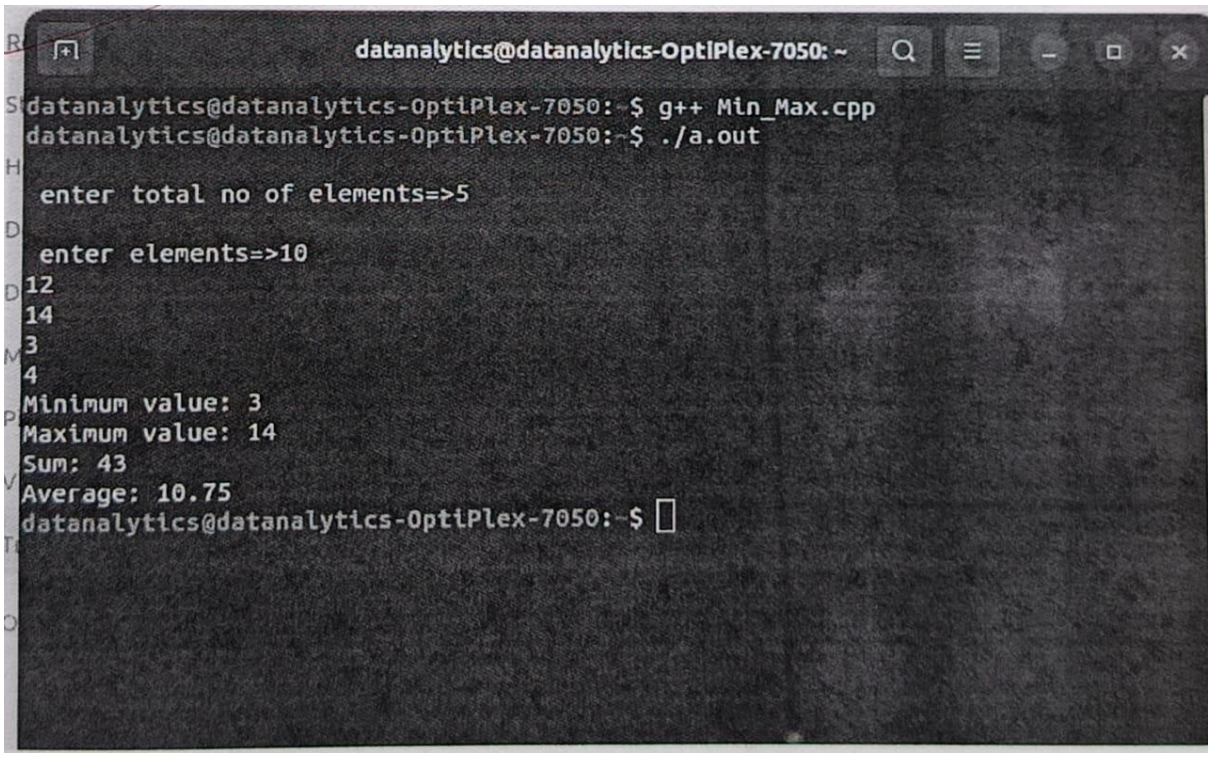
```
#include <iostream>
//#include <vector>
#include <omp.h>
#include <climits>
using namespace std;
void min_reduction(int arr[], int n) {
int min_value = INT_MAX;
#pragma omp parallel for reduction (min: min_value)
for (int i = 0; i < n; i++) {
 if (arr[i] < min_value) {
min_value = arr[i]; }
}
}
cout << "Minimum value: " << min_value << endl;
}
void max_reduction(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 << "Maximum value: " << max_value << endl;
}

void sum_reduction(int arr[], int n) {
int sum = 0;
#pragma omp parallel for reduction(+: sum)
```

```
for (int i = 0; i < n; i++) {
sum += arr[i];
}
cout << "Sum: " << sum << endl;
}
void average_reduction(int arr[], int n) {
int sum = 0;
#pragma omp parallel for reduction(+: sum)
for (int i = 0; i < n; i++) {
sum += arr[i];
}
cout << "Average: " << (double)sum / (n-1) << endl;
}
int main() {
int *arr,n;
cout<<"\n enter total no of elements=>";
cin>>n;
arr=new int[n];
cout<<"\n enter elements=>";
for(int i=0;i<n;i++)
{
cin>>arr[i];
)
// int arr[] = (5, 2, 9, 1, 7, 6, 8, 3, 4);
// int n = size(arr);

min_reduction(arr, n);
max_reduction(arr, n);
sum_reduction(arr, n);
average_reduction(arr, n);
}
```

Output:


datanalytics@datanalytics-OptiPlex-7050: ~
datanalytics@datanalytics-OptiPlex-7050:~$ g++ Min_Max.cpp
datanalytics@datanalytics-OptiPlex-7050:~$ ./a.out
enter total no of elements=>5
enter elements=>10
12
14
3
4
Minimum value: 3
Maximum value: 14
Sum: 43
Average: 10.75
datanalytics@datanalytics-OptiPlex-7050:~$