MinMaxsum code:

```
#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

```
enter total no of elements=>5


 enter elements=>3
21
23
23
24
Minimum value: 3
Maximum value: 24
Sum: 94
Average: 23.5


--------------------------------
Process exited after 10.86 seconds with return value 0
Press any key to continue . . .
```