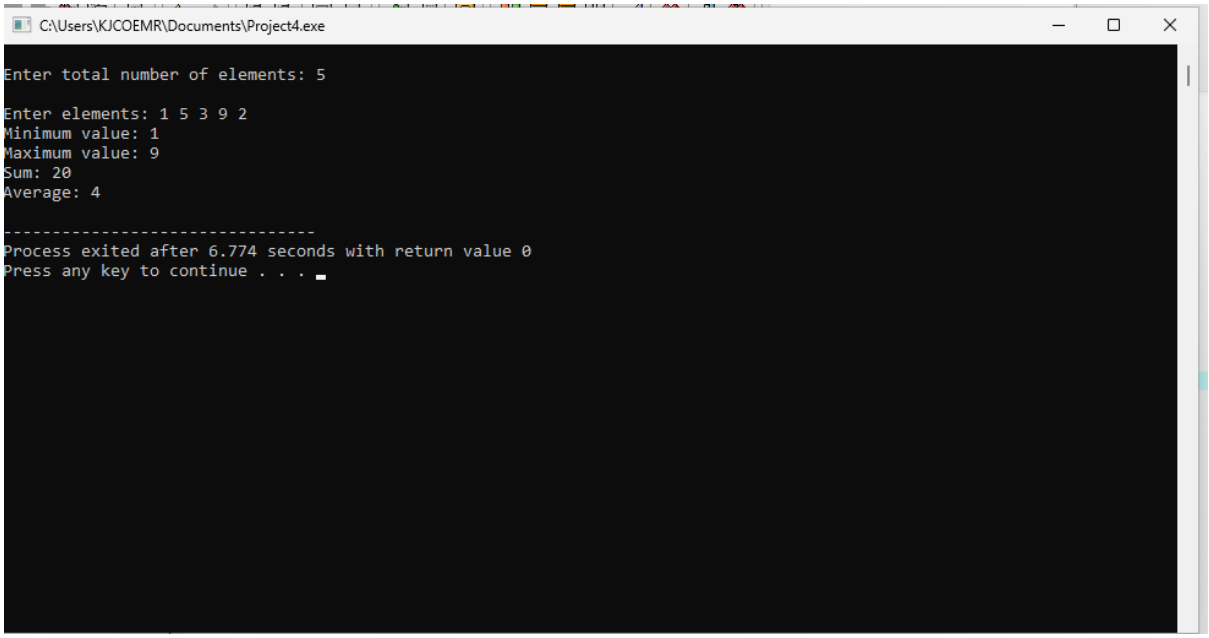
```
#include <iostream>
#include <omp.h>
#include <climits>
using namespace std;
void min_reduction(int arr[], int n) {
  int min_value = INT_MAX;
  #pragma omp parallel for reduction(min: min_value)
  for (int i = 0; i < n; i++) {
    if (arr[i] < min_value) {
      min_value = arr[i];
    }
  }
  cout << "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 << endl;  // Corrected division by n
}
int main() {
  int *arr, n;
  cout << "\nEnter total number of elements: ";
  cin >> n;
  arr = new int[n];

  cout << "\nEnter elements: ";
  for (int i = 0; i < n; i++) {
    cin >> arr[i];
  }
  min_reduction(arr, n);
  max_reduction(arr, n);
  sum_reduction(arr, n);
  average_reduction(arr, n);
  delete[] arr;  // Deallocate memory to avoid memory leak
  return 0;
}
```

Output:


C:\Users\KJCOEMR\Documents\Project4.exe
Enter total number of elements: 5
Enter elements: 1 5 3 9 2
Minimum value: 1
Maximum value: 9
Sum: 20
Average: 4

Process exited after 6.774 seconds with return value 0
Press any key to continue . . .