MIN,MAX,AVG,SUM

```
g++ -fopenmp -o output filename.cpp
./output
```

1

```
#include<iostream>

#include<omp.h>


using namespace std;

int minval(int arr[], int n){

  int minval = arr[0];

  #pragma omp parallel for reduction(min : minval)

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

      if(arr[i] < minval) minval = arr[i];

    }

  return minval;

}


int maxval(int arr[], int n){

  int maxval = arr[0];

  #pragma omp parallel for reduction(max : maxval)

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

      if(arr[i] > maxval) maxval = arr[i];

    }

  return maxval;

}


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

  int sum = 0;

  #pragma omp parallel for reduction(+ : sum)

    for(int i = 0; i < n; i++){
      sum += arr[i];
    }
  return sum;
}

int average(int arr[], int n){
  return (double)sum(arr, n) / n;
}

int main(){
  int n = 5;
  int arr[] = {1,2,3,4,5};
  cout << "The minimum value is: " << minval(arr, n) << '\n';
  cout << "The maximum value is: " << maxval(arr, n) << '\n';
  cout << "The summation is: " << sum(arr, n) << '\n';
  cout << "The average is: " << average(arr, n) << '\n';
  return 0;
}
```

NORMAL FUNCTIONS

```
#include <iostream>
using namespace std;

// Function to find minimum element
int getMin(int arr[], int n)
{
    int res = arr[0];
    for (int i = 1; i < n; i++)
        res = min(res, arr[i]);
    return res;
}

// Function to find maximum element
int getMax(int arr[], int n)
{
    int res = arr[0];
    for (int i = 1; i < n; i++)
```

```
        res = max(res, arr[i]);
    return res;
}

// Function to get Sum
int findSum(int arr[], int n)
{
    int min = getMin(arr, n);
    int max = getMax(arr, n);

    return min + max;
}
```

INT MAIN:

```
start_time = omp_get_wtime();

  mergeSort(arr, 0, n - 1);

  end_time = omp_get_wtime();

  cout << "Time taken by sequential algorithm: " << end_time - start_time << " seconds\n";
```