```
#include <iostream> // --> Includes input-output stream library for cin and cout
#include <omp.h> // --> Includes OpenMP library for parallel processing
#include <climits> // --> Includes limits of integral types like INT_MAX and INT_MIN
using namespace std; // --> Uses standard namespace to avoid prefixing std::

void min_reduction(int arr[], int n) // --> Function to find minimum value using parallel reduction
{
  int min_value = INT_MAX; // --> Initialize min_value with maximum possible integer
#pragma omp parallel for reduction(min : min_value) // --> Parallel for loop with min reduction on min_value
  for (int i = 0; i < n; i++) // --> Loop through array elements
  {
    if (arr[i] < min_value) // --> Check if current element is less than min_value
    {
      min_value = arr[i]; // --> Update min_value if condition is true
    }
  }
  cout << "Minimum value: " << min_value << endl; // --> Output the minimum value
}

void max_reduction(int arr[], int n) // --> Function to find maximum value using parallel reduction
{
  int max_value = INT_MIN; // --> Initialize max_value with minimum possible integer
#pragma omp parallel for reduction(max : max_value) // --> Parallel for loop with max reduction on max_value
  for (int i = 0; i < n; i++) // --> Loop through array elements
  {
    if (arr[i] > max_value) // --> Check if current element is greater than max_value
    {
      max_value = arr[i]; // --> Update max_value if condition is true
    }
```

```
    }
    cout << "Maximum value: " << max_value << endl; // --> Output the maximum value
}

void sum_reduction(int arr[], int n) // --> Function to calculate sum using parallel reduction
{
    int sum = 0; // --> Initialize sum to 0
#pragma omp parallel for reduction(+ : sum) // --> Parallel for loop with addition reduction on sum
    for (int i = 0; i < n; i++) // --> Loop through array elements
    {
        sum += arr[i]; // --> Add current element to sum
    }
    cout << "Sum: " << sum << endl; // --> Output the sum
}

void average_reduction(int arr[], int n) // --> Function to calculate average using parallel reduction
{
    if (n <= 1) // --> Check if array has 1 or fewer elements
    {
        cout << "Average: Cannot calculate (array size too small)" << endl; // --> Output error message for invalid size
        return; // --> Exit the function early
    }

    int sum = 0; // --> Initialize sum to 0
#pragma omp parallel for reduction(+ : sum) // --> Parallel for loop with addition reduction on sum
    for (int i = 0; i < n; i++) // --> Loop through array elements
    {
        sum += arr[i]; // --> Add current element to sum
```

```
    }
    cout << "Average: " << static_cast<double>(sum) / n << endl; // --> Calculate and print average
}

int main() // --> Main function execution starts here
{
    cout << "\n\nName: Shriharsh Deshmukh\nRoll No.62 \t Div.A\n\n"; // --> Display name and roll number

    int *arr, n; // --> Declare pointer for array and variable for size

    cout << "\nEnter total number of elements: "; // --> Prompt user to enter number of elements
    cin >> n; // --> Read number of elements from user

    if (n <= 0) // --> Check if entered number is less than or equal to 0
    {
        cerr << "Error: Array size must be positive" << endl; // --> Show error message if invalid input
        return 1; // --> Exit with error code
    }

    arr = new int[n]; // --> Dynamically allocate memory for array
    cout << "\nEnter elements:\n"; // --> Prompt user to enter array elements
    for (int i = 0; i < n; i++) // --> Loop to input all array elements
    {
        cin >> arr[i]; // --> Read each element into array
    }

    min_reduction(arr, n); // --> Call function to find and display minimum value
    max_reduction(arr, n); // --> Call function to find and display maximum value
    sum_reduction(arr, n); // --> Call function to calculate and display sum
    average_reduction(arr, n); // --> Call function to calculate and display average
```

```
    delete[] arr; // --> Free dynamically allocated memory
    return 0; // --> Return 0 to indicate successful program termination
}

// Run Commands: // --> Commands to compile and run the program
// g++ -fopenmp -o parallel_reduction 5_Min_Max_Sum_Avg_using_Parallel_Reduction.cpp // --> Compile with OpenMP flag
// .\parallel_reduction // --> Run the compiled executable
```