

Assignment 3

Parallel Reduction Implementation:

```
#include <iostream>

#include <vector>

#include <omp.h>

#include <climits>

using namespace std;

void min_reduction(vector<int>& arr) {
    int min_value = INT_MAX;
    #pragma omp parallel for reduction(min: min_value)
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] < min_value) {
            min_value = arr[i];
        }
    }
    cout << "Minimum value: " << min_value << endl;
}

void max_reduction(vector<int>& arr) {
    int max_value = INT_MIN;
    #pragma omp parallel for reduction(max: max_value)
    for (int i = 0; i < arr.size(); i++) {
        if (arr[i] > max_value) {
            max_value = arr[i];
        }
    }
}
```

```

    }
}

cout << "Maximum value: " << max_value << endl;
}

```

```

void sum_reduction(vector<int>& arr) {
    int sum = 0;

    #pragma omp parallel for reduction(+: sum)
    for (int i = 0; i < arr.size(); i++) {
        sum += arr[i];
    }

    cout << "Sum: " << sum << endl;
}

```

```

void average_reduction(vector<int>& arr) {
    int sum = 0;

    #pragma omp parallel for reduction(+: sum)
    for (int i = 0; i < arr.size(); i++) {
        sum += arr[i];
    }

    cout << "Average: " << (double)sum / arr.size() << endl;
}

```

```

int main() {
    vector<int> arr;

    arr.push_back(5);
    arr.push_back(2);
    arr.push_back(9);
    arr.push_back(1);
}

```

```
arr.push_back(7);  
arr.push_back(6);  
arr.push_back(8);  
arr.push_back(3);  
arr.push_back(4);  
  
min_reduction(arr);  
max_reduction(arr);  
sum_reduction(arr);  
average_reduction(arr);  
}
```

OUTPUT:

Minimum value: 1

Maximum value: 9

Sum: 45

Average: 5