

Experiment 3

Implement Min, Max, Sum and Average operations using Parallel Reduction

1. max.c

```
#include <stdio.h>
#include <omp.h>
int main()
{
    double arr[10];
    omp_set_num_threads(4);
    double max_val=0.0;
    int i;
    for( i=0; i<10; i++)
        arr[i] = 2.0 + i;
    #pragma omp parallel for reduction(max : max_val)
    for( i=0; i<10; i++)
    {
        printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
        if(arr[i] > max_val)
        {
            max_val = arr[i];
        }
    }
    printf("\nmax_val = %f", max_val);
}
```

Output:

```
user1@user1-ThinkCentre-E73:~$ g++ max.c -fopenmp
```

```
user1@user1-ThinkCentre-E73:~$ ./a.out
```

```
thread id = 3 and i = 8
```

```
thread id = 3 and i = 9
```

```
thread id = 0 and i = 0
```

```
thread id = 0 and i = 1
```

```
thread id = 0 and i = 2
```

```
thread id = 1 and i = 3
```

thread id = 1 and i = 4

thread id = 1 and i = 5

thread id = 2 and i = 6

thread id = 2 and i = 7

max_val = 11.000000

```
C max.c > main()
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main()
5  {
6      double arr[10];
7      omp_set_num_threads(4);
8      double max_val=0.0;
9      int i;
10     for( i=0; i<10; i++)
11         arr[i] = 2.0 + i;
12
13     #pragma omp parallel for reduction(max : max_val)
14     for( i=0; i<10; i++)
15     {
16         printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
17         if(arr[i] > max_val)
18         {
19             max_val = arr[i];
20         }
21     }
22
23     printf("\nmax_val = %f", max_val);
24 }
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ max.c -fopenmp
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
```

thread id = 3 and i = 8

thread id = 3 and i = 9

thread id = 2 and i = 6

thread id = 2 and i = 7

thread id = 1 and i = 3

thread id = 1 and i = 4

thread id = 1 and i = 5

thread id = 0 and i = 0

thread id = 0 and i = 1

thread id = 0 and i = 2

max_val = 11.000000

2. min.c

```
#include <stdio.h>
#include <omp.h>

int main()
{
    double arr[10];
    omp_set_num_threads(4);
    double min_val=9.0;
    int i;

    for( i=0; i<10; i++)
        arr[i] = 2.0 + i;

    #pragma omp parallel for reduction(min : min_val)
    for( i=0;i<10; i++)
    {
        printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
        if(arr[i] < min_val)
        {
            min_val = arr[i];
        }
    }
    printf("\nmin_val = %f", min_val);
}
```

Output:

user1@user1-ThinkCentre-E73:~\$ g++ min.c -fopenmp

user1@user1-ThinkCentre-E73:~\$./a.out

thread id = 2 and i = 6

thread id = 2 and i = 7

thread id = 0 and i = 0

thread id = 0 and i = 1

thread id = 0 and i = 2

thread id = 3 and i = 8

thread id = 3 and i = 9

thread id = 1 and i = 3

thread id = 1 and i = 4

thread id = 1 and i = 5

```
C min.c > main()
1  #include <stdio.h>
2  #include <omp.h>
3
4  int main()
5  {
6      double arr[10];
7      omp_set_num_threads(4);
8      double min_val=9.0;
9      int i;
10     for( i=0; i<10; i++)
11     |   arr[i] = 2.0 + i;
12     #pragma omp parallel for reduction(min : min_val)
13     for( i=0;i<10; i++)
14     {
15         printf("thread id = %d and i = %d \n", omp_get_thread_num(),i);
16         if(arr[i] < min_val)
17         {
18             min_val = arr[i];
19         }
20     }
21     printf("\nmin_val = %f", min_val);
22 }
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ min.c -fopenmp
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
```

thread id = 0 and i = 0

thread id = 0 and i = 1

thread id = 0 and i = 2

thread id = 2 and i = 6

thread id = 2 and i = 7

thread id = 1 and i = 3

thread id = 1 and i = 4

thread id = 1 and i = 5

thread id = 3 and i = 8

thread id = 3 and i = 9

min_val = 2.000000

3. sum.c

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{
    int i, n;
    float a[100], b[100], sum;

    /* Some initializations */
    n = 3;
    for (i=0; i < n; i++)
        a[i] = b[i] = i * 1.0;
    sum = 0.0;

    #pragma omp parallel for reduction(+:sum)
    for (i=0; i < n; i++)
        sum = sum + (a[i] * b[i]);

    printf(" Sum = %f\n",sum);
}
```

Output:

user1@user1-ThinkCentre-E73:~\$ g++ sum.c -fopenmp

user1@user1-ThinkCentre-E73:~\$./a.out

Sum = 5.000000

C sum.c > ...

```
1  ✓ #include <omp.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  ✓ int main (int argc, char *argv[])
6  {
7      int i, n;
8      float a[100], b[100], sum;
9
10     /* Some initializations */
11     n = 3;
12     for (i=0; i < n; i++)
13     |   a[i] = b[i] = i * 1.0;
14     sum = 0.0;
15
16     #pragma omp parallel for reduction(+:sum)
17     for (i=0; i < n; i++)
18     |   sum = sum + (a[i] * b[i]);
19
20     printf(" Sum = %f\n",sum);
21 }
22
```

PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ sum.c -fopenmp

PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a

Sum = 5.000000

4. avg.cpp

```
#include<iostream>
#include<omp.h>
using namespace std;

int main()
{
    int a[100],n,i;
    cout<<"enter the number of elements in array: ";
    cin>>n;
    cout<<"\nenter array elements : ";
    for(i=0;i<n;i++)
    {
        cin>>a[i];
    }
    cout<<"\narray elements are:\t";
    for(i=0;i<n;i++)
    {
        cout<<a[i]<<"\t";
    }
    float avg=0,sum=0;
    #pragma omp parallel
    {
        int id=omp_get_thread_num();
        #pragma omp for
        for(i=0;i<n;i++)
        {
            sum=sum+a[i];
            cout<<"\nfor i = " <<i<<" thread " <<id<<" is executing "<<endl;
        }
    }
    avg=sum/n;
    cout<<"output = "<<avg<<endl;
}
```

Output:

enter the number of elements in array : 5

enter array elements : 3 4 6 7 8

array elements are: 3 4 6 7 8

for i= 0 thread 0 is executing

for i= 2 thread 1 is executing

for i= 3 thread 2 is executing

for i= 4 thread 3 is executing

for i= 1 thread 0 is executing

output = 3.4

```
avg.cpp > main()
1  #include<iostream>
2  #include<omp.h>
3  using namespace std;
4
5  int main()
6  {
7      int a[100],n,i;
8      cout<<"enter the number of elements in array: ";
9      cin>>n;
10     cout<<"\nenter array elements : ";
11     for(i=0;i<n;i++)
12     {
13         cin>>a[i];
14     }
15     cout<<"\narray elements are:\t";
16     for(i=0;i<n;i++)
17     {
18         cout<<a[i]<<"\t";
19     }
20     float avg=0,sum=0;
21     #pragma omp parallel
22     {
23         int id=omp_get_thread_num();
24         #pragma omp for
25         for(i=0;i<n;i++)
26         {
27             sum=sum+a[i];
28             cout<<"\nfor i = " <<i<<" thread "<<id<<" is executing "<<endl;
29         }
30     }
31     avg=sum/n;
32     cout<<"output = " <<avg<<endl;
33 }
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> g++ avg.cpp -fopenmp
```

```
PS C:\Users\Admin\Desktop\BE\Practicals\HPC> ./a
```

enter the number of elements in array: 3

enter array elements : 2 3 4

array elements are: 2 3 4

for i = 2 thread 2 is executing

for i = 1 thread 1 is executing

for i = 0 thread 0 is executing

output = 3

-