National Institute of Technology Karnataka Surathkal,
Mangalore - 575025

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB ASSIGNMENT 4

Submitted for Parallel Computing (IT301) By

Gaurav Chaurasia 181CV155

To

Dr. Geetha V *Dept of IT, NITK Surathkal*

Code link
github

problem - 1

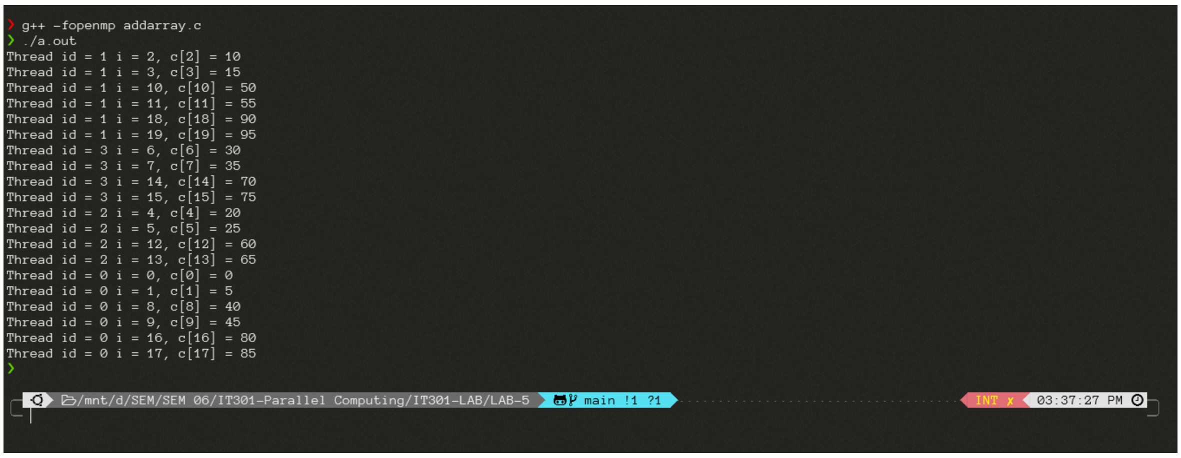
```
#include <stdio.h>
#include <omp.h>
int main()
{
    int i, n, chunk;
    int a[20], b[20], c[20];
    n = 20;
    chunk = 2;
    /*initializing array*/
    for (i = 0; i < n; i++)
    {
        a[i] = i * 2;
        b[i] = i * 3;
    }
#pragma omp parallel for default(shared) private(i)
schedule(static, chunk)
    for (i = 0; i < n; i++)
    {
        c[i] = a[i] + b[i];
        printf("Thread id = %d i = %d, c[%d] = %d\n",
omp_get_thread_num(), i, i, c[i]);
    }
}
```

Output

```
❯ g++ -fopenmp addarray.c
❯ ./a.out
```

Chunk = 2 & threads = 4

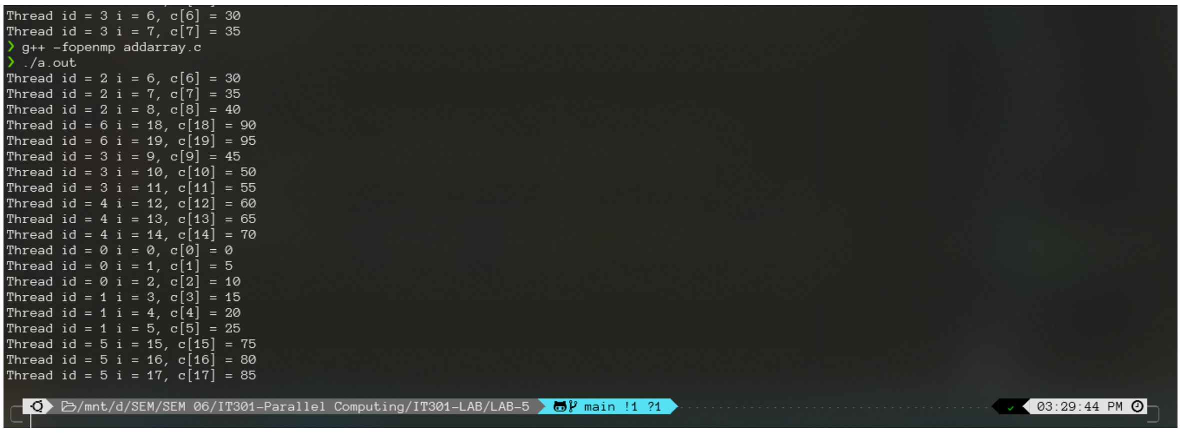
```
Thread id = 1 i = 2, c[2] = 10
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 10, c[10] = 50
Thread id = 1 i = 11, c[11] = 55
Thread id = 1 i = 18, c[18] = 90
Thread id = 1 i = 19, c[19] = 95
Thread id = 3 i = 6, c[6] = 30
Thread id = 3 i = 7, c[7] = 35
Thread id = 3 i = 14, c[14] = 70
Thread id = 3 i = 15, c[15] = 75
Thread id = 2 i = 4, c[4] = 20
Thread id = 2 i = 5, c[5] = 25
Thread id = 2 i = 12, c[12] = 60
Thread id = 2 i = 13, c[13] = 65
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 8, c[8] = 40
Thread id = 0 i = 9, c[9] = 45
Thread id = 0 i = 16, c[16] = 80
Thread id = 0 i = 17, c[17] = 85
```



```
❯ g++ -fopenmp addarray.c
❯ ./a.out
Thread id = 1 i = 2, c[2] = 10
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 10, c[10] = 50
Thread id = 1 i = 11, c[11] = 55
Thread id = 1 i = 18, c[18] = 90
Thread id = 1 i = 19, c[19] = 95
Thread id = 3 i = 6, c[6] = 30
Thread id = 3 i = 7, c[7] = 35
Thread id = 3 i = 14, c[14] = 70
Thread id = 3 i = 15, c[15] = 75
Thread id = 2 i = 4, c[4] = 20
Thread id = 2 i = 5, c[5] = 25
Thread id = 2 i = 12, c[12] = 60
Thread id = 2 i = 13, c[13] = 65
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 8, c[8] = 40
Thread id = 0 i = 9, c[9] = 45
Thread id = 0 i = 16, c[16] = 80
Thread id = 0 i = 17, c[17] = 85
❯

/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-5  main !1 ?1 ··········· INT ✗  03:37:27 PM
```

Chunk = 2 & threads = 2

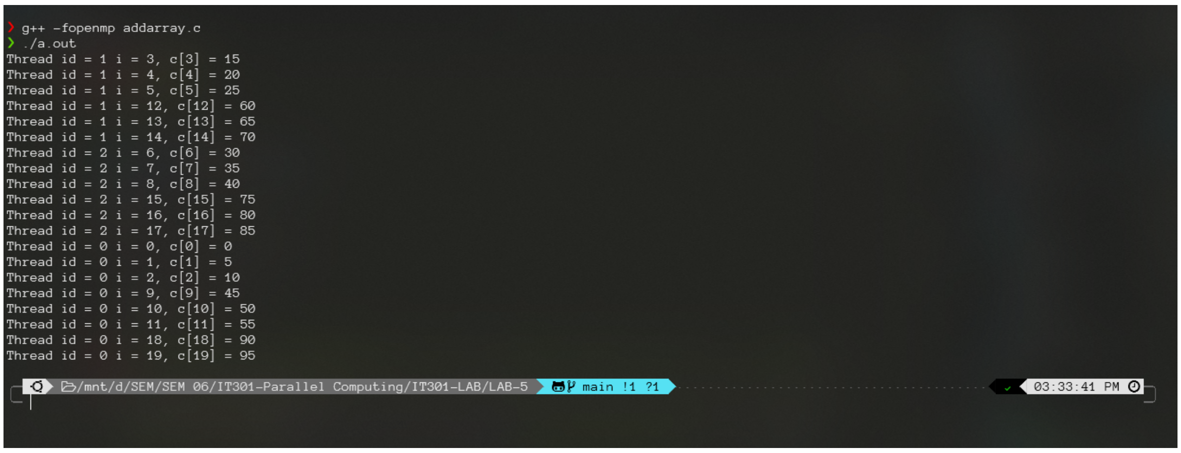
```
❯ g++ -fopenmp addarray.c
❯ ./a.out
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 4, c[4] = 20
Thread id = 0 i = 5, c[5] = 25
Thread id = 0 i = 8, c[8] = 40
Thread id = 0 i = 9, c[9] = 45
Thread id = 0 i = 12, c[12] = 60
Thread id = 0 i = 13, c[13] = 65
Thread id = 0 i = 16, c[16] = 80
Thread id = 0 i = 17, c[17] = 85
Thread id = 1 i = 2, c[2] = 10
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 6, c[6] = 30
Thread id = 1 i = 7, c[7] = 35
Thread id = 1 i = 10, c[10] = 50
Thread id = 1 i = 11, c[11] = 55
Thread id = 1 i = 14, c[14] = 70
Thread id = 1 i = 15, c[15] = 75
Thread id = 1 i = 18, c[18] = 90
Thread id = 1 i = 19, c[19] = 95
```



```
Thread id = 3 i = 6, c[6] = 30
Thread id = 3 i = 7, c[7] = 35
❯ g++ -fopenmp addarray.c
❯ ./a.out
Thread id = 2 i = 6, c[6] = 30
Thread id = 2 i = 7, c[7] = 35
Thread id = 2 i = 8, c[8] = 40
Thread id = 6 i = 18, c[18] = 90
Thread id = 6 i = 19, c[19] = 95
Thread id = 3 i = 9, c[9] = 45
Thread id = 3 i = 10, c[10] = 50
Thread id = 3 i = 11, c[11] = 55
Thread id = 4 i = 12, c[12] = 60
Thread id = 4 i = 13, c[13] = 65
Thread id = 4 i = 14, c[14] = 70
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 2, c[2] = 10
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 4, c[4] = 20
Thread id = 1 i = 5, c[5] = 25
Thread id = 5 i = 15, c[15] = 75
Thread id = 5 i = 16, c[16] = 80
Thread id = 5 i = 17, c[17] = 85
/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-5   main !1 ?1      03:29:44 PM
```

Chunk = 3 & threads = 3

```
❯ g++ -fopenmp addarray.c
❯ ./a.out
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 4, c[4] = 20
Thread id = 1 i = 5, c[5] = 25
Thread id = 1 i = 12, c[12] = 60
Thread id = 1 i = 13, c[13] = 65
Thread id = 1 i = 14, c[14] = 70
Thread id = 2 i = 6, c[6] = 30
Thread id = 2 i = 7, c[7] = 35
Thread id = 2 i = 8, c[8] = 40
Thread id = 2 i = 15, c[15] = 75
Thread id = 2 i = 16, c[16] = 80
Thread id = 2 i = 17, c[17] = 85
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 2, c[2] = 10
Thread id = 0 i = 9, c[9] = 45
Thread id = 0 i = 10, c[10] = 50
Thread id = 0 i = 11, c[11] = 55
Thread id = 0 i = 18, c[18] = 90
Thread id = 0 i = 19, c[19] = 95
```



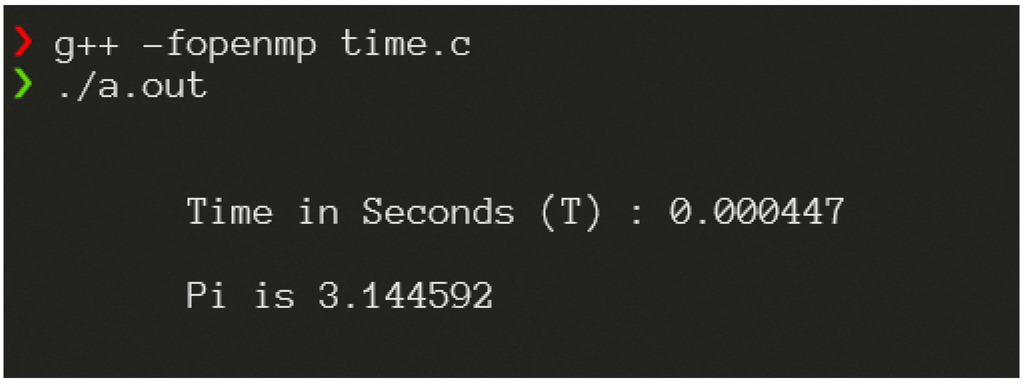
```
❯ g++ -fopenmp addarray.c
❯ ./a.out
Thread id = 1 i = 3, c[3] = 15
Thread id = 1 i = 4, c[4] = 20
Thread id = 1 i = 5, c[5] = 25
Thread id = 1 i = 12, c[12] = 60
Thread id = 1 i = 13, c[13] = 65
Thread id = 1 i = 14, c[14] = 70
Thread id = 2 i = 6, c[6] = 30
Thread id = 2 i = 7, c[7] = 35
Thread id = 2 i = 8, c[8] = 40
Thread id = 2 i = 15, c[15] = 75
Thread id = 2 i = 16, c[16] = 80
Thread id = 2 i = 17, c[17] = 85
Thread id = 0 i = 0, c[0] = 0
Thread id = 0 i = 1, c[1] = 5
Thread id = 0 i = 2, c[2] = 10
Thread id = 0 i = 9, c[9] = 45
Thread id = 0 i = 10, c[10] = 50
Thread id = 0 i = 11, c[11] = 55
Thread id = 0 i = 18, c[18] = 90
Thread id = 0 i = 19, c[19] = 95
/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-5   main !1 ?1      03:33:41 PM
```

Problem 2: compare sequential and parallel program execution times

```
#include <stdio.h>
#include <sys/time.h>
#include <omp.h>
#include <stdlib.h>
int main(void)
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;
    double pi, x;
    int i, N;
    pi = 0.0;
    N = 1000;
    gettimeofday(&TimeValue_Start, &TimeZone_Start);
#pragma omp parallel for private(x) reduction(+ : pi)
    for (i = 0; i <= N; i++)
    {
        x = (double)i / N;
        pi += 4 / (1 + x * x);
    }
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 +
TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 +
TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    printf("\n\n\tTime in Seconds (T) : %lf\n", time_overhead);
    pi = pi / N;
    printf("\n \tPi is %f\n\n", pi);
}
```

Output

```
❯ g++ -fopenmp time.c
❯ ./a.out
parallel
        Time in Seconds (T) : 0.000447

        Pi is 3.144592
```



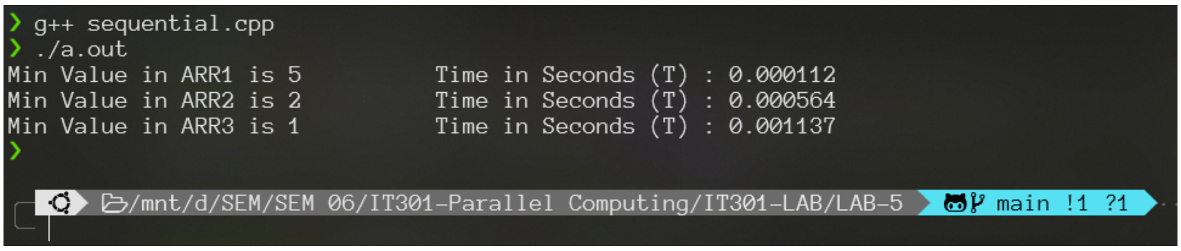
```
❯ g++ -fopenmp time.c
❯ ./a.out


        Time in Seconds (T) : 0.000447

        Pi is 3.144592
```

Problem 3 Write a sequential program to find the smallest element in an array. Convert the same program for parallel execution

```
/* sequential */
#include <iostream>
#include <stdio.h>
#include <climits>
#include <vector>
#include <sys/time.h>

using namespace std;
#define SIZE 20

void get_randome_array(vector<int> &arr) {
    for (int i = 0; i < arr.size(); i++) {
        arr[i] = 1 + (rand() % 100000);
    }
}

int min_value(const vector<int> &arr) {
    int mint = INT_MAX;
    for (auto it: arr) {
        mint = min(mint, it);
    }
    return mint;
}

int main() {
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;
    vector<int> ARR1(10000), ARR2(50000), ARR3(100000);

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    get_randome_array(ARR1);
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in ARR1 is " << min_value(ARR1) << "\t\tTime in Seconds
(T) : " << time_overhead << endl;

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    get_randome_array(ARR2);
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in ARR2 is " << min_value(ARR2) << "\t\tTime in Seconds
(T) : " << time_overhead << endl;

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    get_randome_array(ARR3);
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in ARR3 is " << min_value(ARR3) << "\t\tTime in Seconds
(T) : " << time_overhead << endl;

    return 0;
}
```

Output

```
❯ g++ sequential.cpp
❯ ./a.out
Min Value in ARR1 is 5          Time in Seconds (T) : 0.000112
Min Value in ARR2 is 2          Time in Seconds (T) : 0.000564
Min Value in ARR3 is 1          Time in Seconds (T) : 0.001137
```



```
❯ g++ sequential.cpp
❯ ./a.out
Min Value in ARR1 is 5          Time in Seconds (T) : 0.000112
Min Value in ARR2 is 2          Time in Seconds (T) : 0.000564
Min Value in ARR3 is 1          Time in Seconds (T) : 0.001137
❯
/mnt/d/SEM/SEM 06/IT301-Parallel Computing/IT301-LAB/LAB-5   main !1 ?1
```

```
/* parallel */
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <omp.h>
#include <sys/time.h>
using namespace std;

int main(void)
{
    struct timeval TimeValue_Start;
    struct timezone TimeZone_Start;
    struct timeval TimeValue_Final;
    struct timezone TimeZone_Final;
    long time_start, time_end;
    double time_overhead;
    int i, a[10000], b[50000], c[100000],sml;
    int tid;
    //initializing array randomly
    for (i = 0; i < 10000; i++)
    {
        a[i] = 1 + (rand() % 100000);
    }
    for (i = 0; i < 50000; i++)
    {
        b[i] = 1 + (rand() % 100000);
    }
    for (i = 0; i < 100000; i++)
    {
        c[i] = 1 + (rand() % 100000);
    }

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    sml = a[0];
#pragma omp parallel private(tid) num_threads(4)
    {
        tid = omp_get_thread_num();
#pragma omp for private(i) schedule(static, 5)
        for (i = 0; i < 10000; ++i)
        {
            if (sml > a[i])
                sml = a[i];
        }
    }
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in a is " << sml << "\t\tTime in Seconds (T) : " <<
time_overhead << endl;

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    sml = b[0];
#pragma omp parallel private(tid) num_threads(4)
    {
        tid = omp_get_thread_num();
#pragma omp for private(i) schedule(static, 5)
        for (i = 0; i < 50000; ++i)
        {
            if (sml > b[i])
                sml = b[i];
        }
    }
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in b is " << sml << "\t\tTime in Seconds (T) : " <<
time_overhead << endl;

    gettimeofday(&TimeValue_Start, &TimeZone_Start);
    sml = c[0];
```

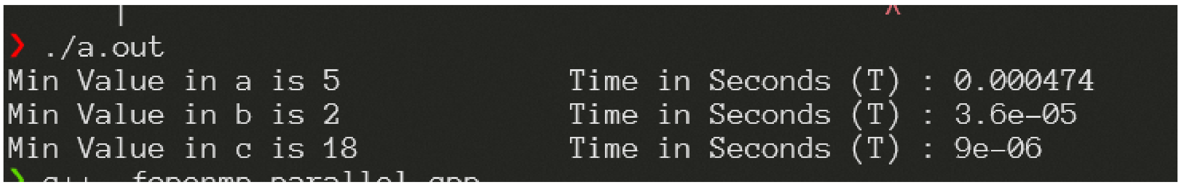
```
#pragma omp parallel private(tid) num_threads(4)
    {
        tid = omp_get_thread_num();
#pragma omp for private(i) schedule(static, 5)
        for (i = 0; i < 10000; ++i)
        {
            if (sml > c[i])
                sml = c[i];
        }
    }
    gettimeofday(&TimeValue_Final, &TimeZone_Final);
    time_start = TimeValue_Start.tv_sec * 1000000 + TimeValue_Start.tv_usec;
    time_end = TimeValue_Final.tv_sec * 1000000 + TimeValue_Final.tv_usec;
    time_overhead = (time_end - time_start) / 1000000.0;
    cout << "Min Value in c is " << sml << "\t\tTime in Seconds (T) : " <<
time_overhead << endl;
}
```

Output

```
❯ g++ -fopenmp parallel.cpp
❯ ./a.out
```

Static, 5

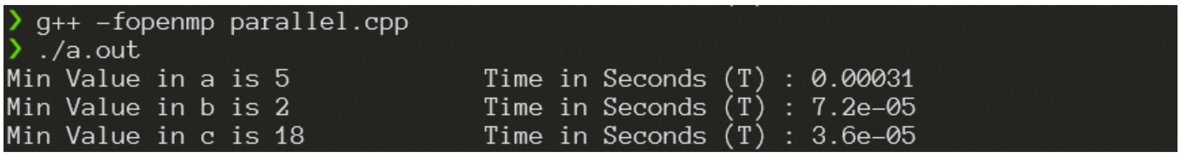
```
Min Value in a is 5             Time in Seconds (T) : 0.000474
Min Value in b is 2             Time in Seconds (T) : 3.6e-05
Min Value in c is 18            Time in Seconds (T) : 9e-06
```



```
❯ ./a.out
Min Value in a is 5             Time in Seconds (T) : 0.000474
Min Value in b is 2             Time in Seconds (T) : 3.6e-05
Min Value in c is 18            Time in Seconds (T) : 9e-06
```

Auto

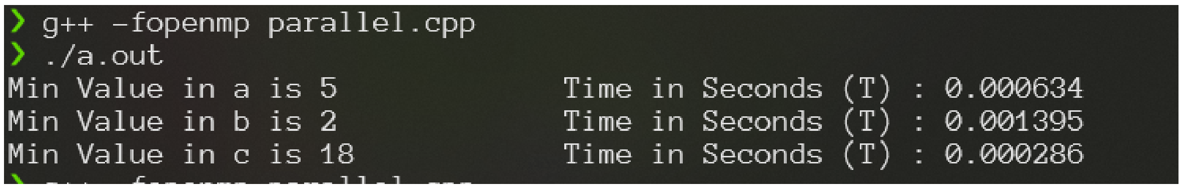
```
Min Value in a is 5             Time in Seconds (T) : 0.00031
Min Value in b is 2             Time in Seconds (T) : 7.2e-05
Min Value in c is 18            Time in Seconds (T) : 3.6e-05
```



```
❯ g++ -fopenmp parallel.cpp
❯ ./a.out
Min Value in a is 5             Time in Seconds (T) : 0.00031
Min Value in b is 2             Time in Seconds (T) : 7.2e-05
Min Value in c is 18            Time in Seconds (T) : 3.6e-05
```

dynamic, 5

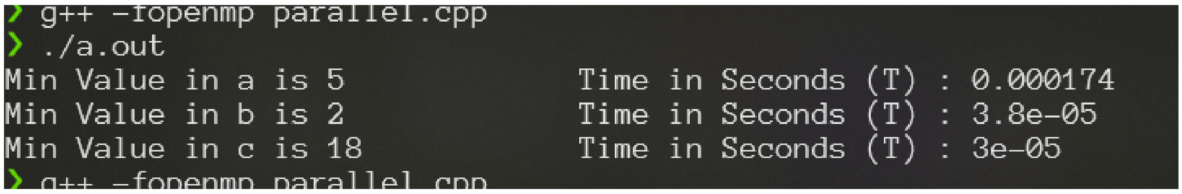
```
Min Value in a is 5             Time in Seconds (T) : 0.000634
Min Value in b is 2           Time in Seconds (T) : 0.001395
Min Value in c is 18          Time in Seconds (T) : 0.000286
```



```
❯ g++ -fopenmp parallel.cpp
❯ ./a.out
Min Value in a is 5          Time in Seconds (T) : 0.000634
Min Value in b is 2          Time in Seconds (T) : 0.001395
Min Value in c is 18         Time in Seconds (T) : 0.000286
```

Guided

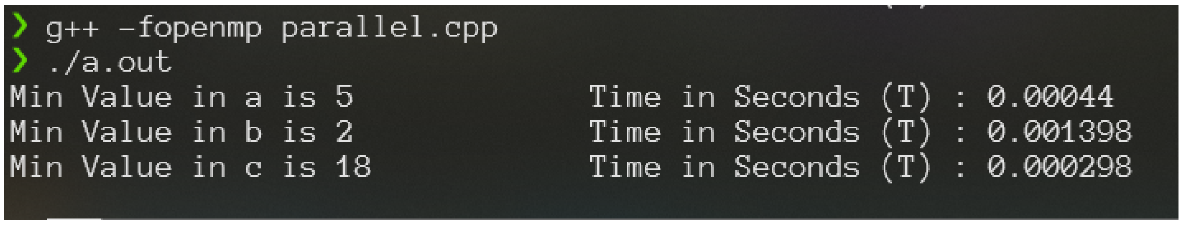
```
Min Value in a is 5           Time in Seconds (T) : 0.000174
Min Value in b is 2           Time in Seconds (T) : 3.8e-05
Min Value in c is 18          Time in Seconds (T) : 3e-05
```



```
❯ g++ -fopenmp parallel.cpp
❯ ./a.out
Min Value in a is 5          Time in Seconds (T) : 0.000174
Min Value in b is 2          Time in Seconds (T) : 3.8e-05
Min Value in c is 18         Time in Seconds (T) : 3e-05
❯ g++ -fopenmp parallel.cpp
```

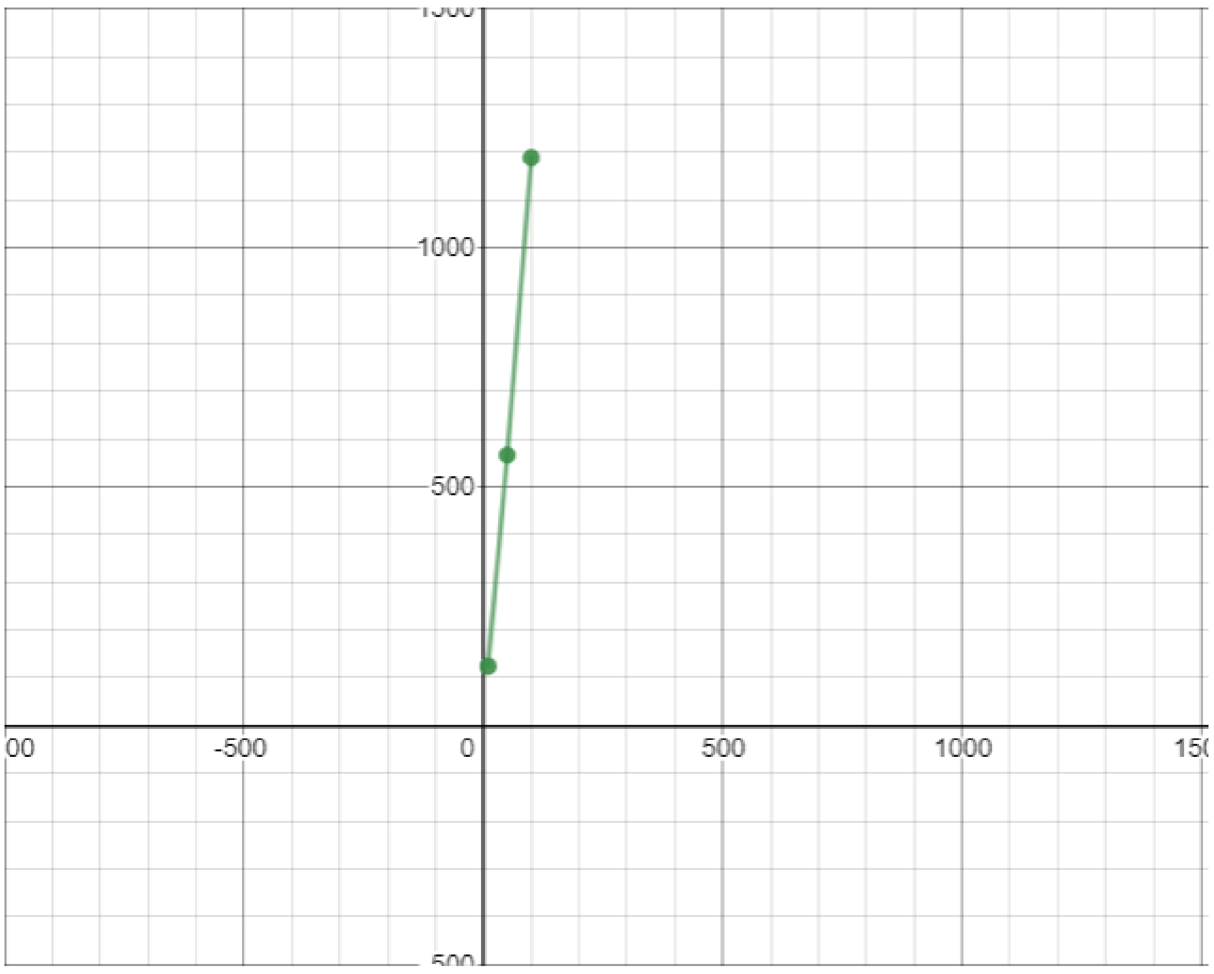
Runtime

```
Min Value in a is 5           Time in Seconds (T) : 0.00044
Min Value in b is 2           Time in Seconds (T) : 0.001398
Min Value in c is 18          Time in Seconds (T) : 0.000298
```



```
❯ g++ -fopenmp parallel.cpp
❯ ./a.out
Min Value in a is 5          Time in Seconds (T) : 0.00044
Min Value in b is 2          Time in Seconds (T) : 0.001398
Min Value in c is 18         Time in Seconds (T) : 0.000298
```

Schedule	Total execution for number of iterations 10K	Total execution for number of iterations 50K	Total execution for number of iterations 100K
Sequential execution	0.000124	0.000566	0.001188
static	0.000268	4.1e-05	1e-05
Static, chunksize	0.000192	3.5e-05	1.4e-05
Dynamic, chunksize	0.000174	3.8e-05	5.3e-05
Guided	0.000174	3.8e-05	3e-05
runtime	0.00044	0.001398	0.000298


1000
500
00
-500
0
500
1000
500