

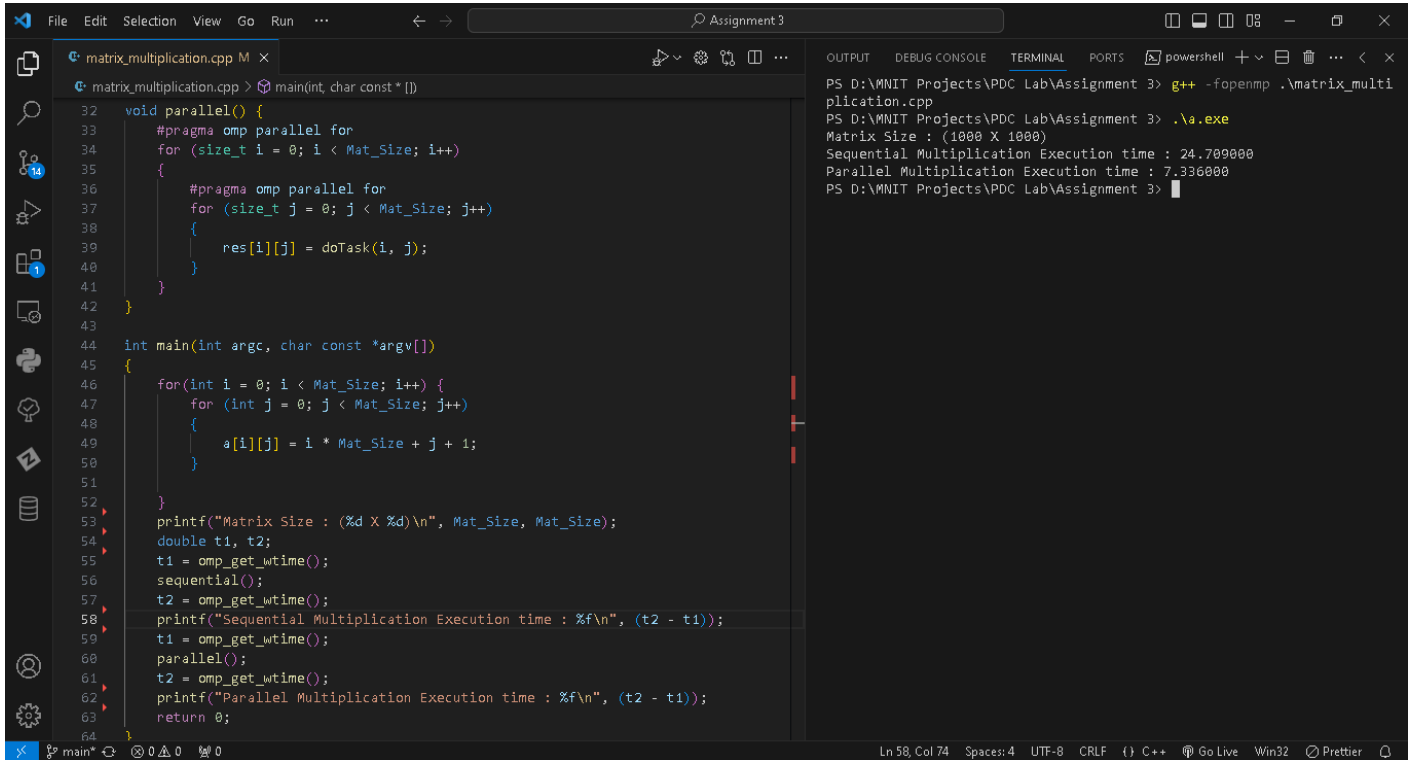
# Assignment 3

Name: Ranjan Baro

ID: 2023PCP5274

1. Write **OpenMP** parallel program for matrix multiplication and compare the performance of serial and parallel version of matrix multiplication.

Ans:



```
matrix_multiplication.cpp M X
main(int, char const* [])
32 void parallel() {
33     #pragma omp parallel for
34     for (size_t i = 0; i < Mat_Size; i++)
35     {
36         #pragma omp parallel for
37         for (size_t j = 0; j < Mat_Size; j++)
38         {
39             res[i][j] = doTask(i, j);
40         }
41     }
42 }
43
44 int main(int argc, char const *argv[])
45 {
46     for(int i = 0; i < Mat_Size; i++) {
47         for (int j = 0; j < Mat_Size; j++)
48         {
49             a[i][j] = i * Mat_Size + j + 1;
50         }
51     }
52     printf("Matrix Size : (%d X %d)\n", Mat_Size, Mat_Size);
53     double t1, t2;
54     t1 = omp_get_wtime();
55     sequential();
56     t2 = omp_get_wtime();
57     printf("Sequential Multiplication Execution time : %f\n", (t2 - t1));
58     t1 = omp_get_wtime();
59     parallel();
60     t2 = omp_get_wtime();
61     printf("Parallel Multiplication Execution time : %f\n", (t2 - t1));
62     return 0;
63 }
64 }
```

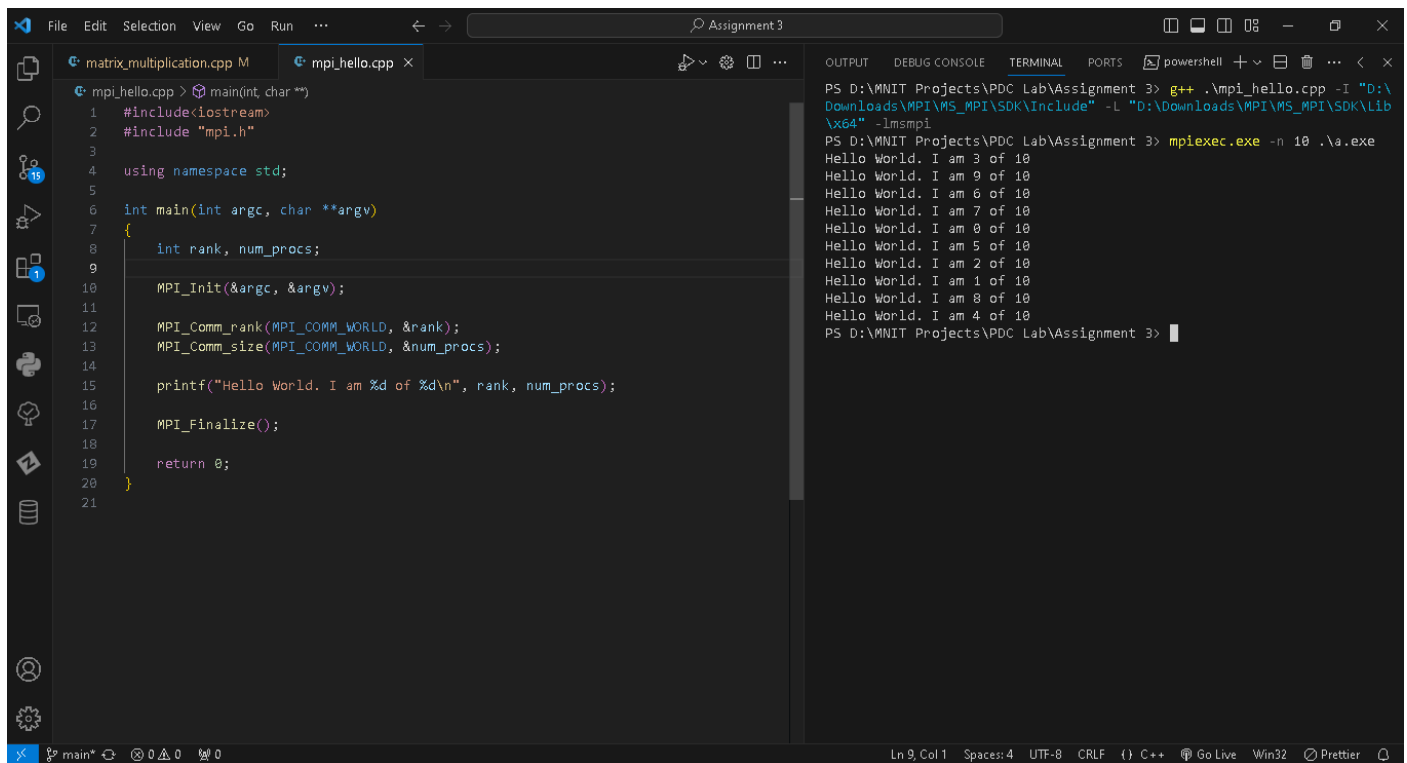
```
PS D:\MNIT Projects\PDC Lab\Assignment 3> g++ -fopenmp .\matrix_multi
plication.cpp
PS D:\MNIT Projects\PDC Lab\Assignment 3> .\a.exe
Matrix Size : (1000 X 1000)
Sequential Multiplication Execution time : 24.709000
Parallel Multiplication Execution time : 7.336000
PS D:\MNIT Projects\PDC Lab\Assignment 3>
```

The time required to calculate matrix multiplication of matrix size using **serial computation** (1000 x 1000) is 24.7090

The time required to calculate matrix multiplication of matrix size using **parallel computation** (1000 x 1000) is 7.3360

2. Write MPI hello world program. This exercise will help you to be familiar with the basic commands (compile and run) and routines of MPI programming model.

Ans.



The screenshot shows the Visual Studio Code editor with a file named `mpi_hello.cpp` open. The code is as follows:

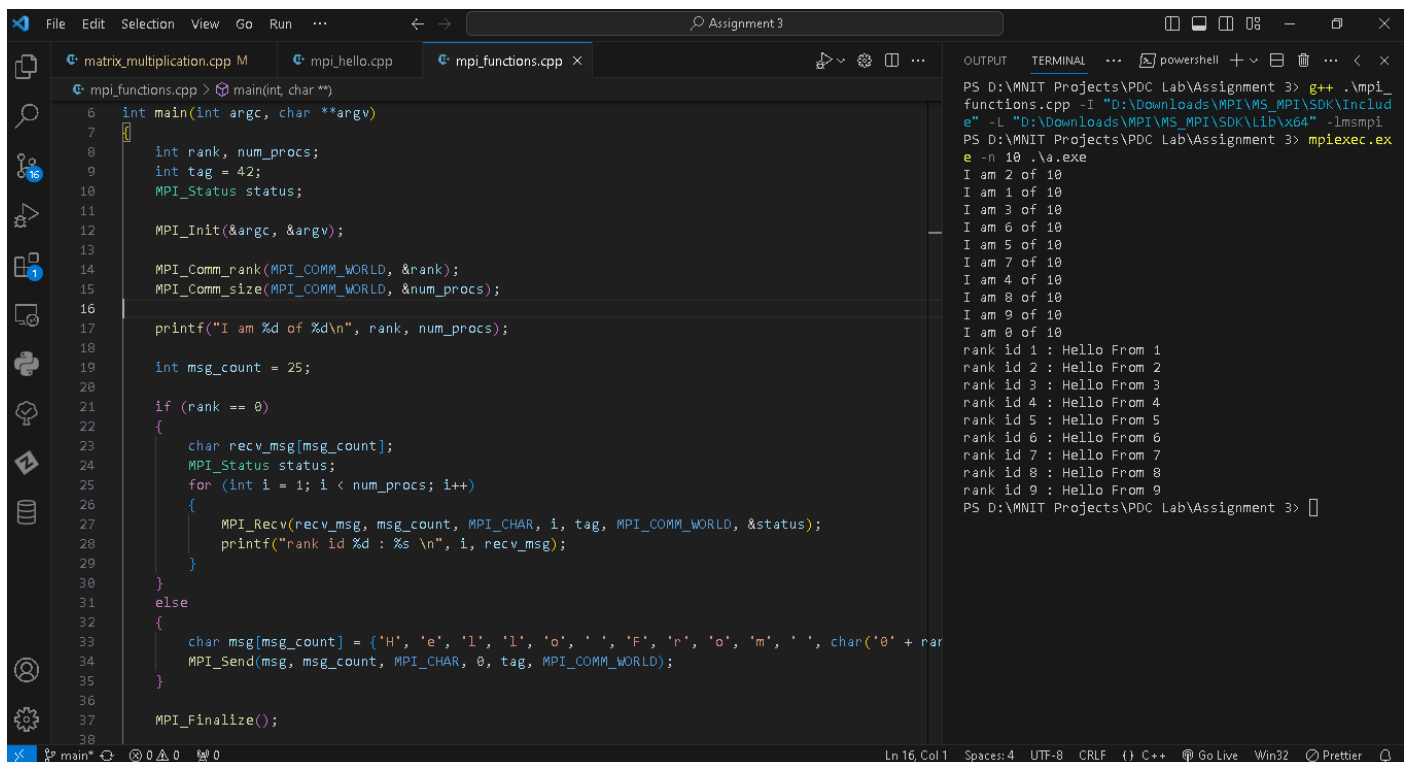
```
1 #include <iostream>
2 #include "mpi.h"
3
4 using namespace std;
5
6 int main(int argc, char **argv)
7 {
8     int rank, num_procs;
9
10    MPI_Init(&argc, &argv);
11
12    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
13    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
14
15    printf("Hello World. I am %d of %d\n", rank, num_procs);
16
17    MPI_Finalize();
18
19    return 0;
20 }
```

The output window on the right shows the command prompt with the following commands and output:

```
PS D:\MNIT Projects\PDC Lab\Assignment 3> g++ .\mpi_hello.cpp -I "D:\Downloads\MPI\MS_MPI\SDK\Include" -L "D:\Downloads\MPI\MS_MPI\SDK\Lib\x64" -lmsmpi
PS D:\MNIT Projects\PDC Lab\Assignment 3> mpiexec.exe -n 10 .\a.exe
Hello World. I am 3 of 10
Hello World. I am 9 of 10
Hello World. I am 6 of 10
Hello World. I am 7 of 10
Hello World. I am 8 of 10
Hello World. I am 5 of 10
Hello World. I am 2 of 10
Hello World. I am 1 of 10
Hello World. I am 0 of 10
Hello World. I am 4 of 10
PS D:\MNIT Projects\PDC Lab\Assignment 3>
```

3. Study about following basic MPI subroutines.

Ans.



The screenshot shows the Visual Studio Code editor with a file named `mpi_functions.cpp` open. The code is as follows:

```
1 #include <iostream>
2 #include "mpi.h"
3
4 using namespace std;
5
6 int main(int argc, char **argv)
7 {
8     int rank, num_procs;
9     int tag = 42;
10    MPI_Status status;
11
12    MPI_Init(&argc, &argv);
13
14    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
15    MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
16
17    printf("I am %d of %d\n", rank, num_procs);
18
19    int msg_count = 25;
20
21    if (rank == 0)
22    {
23        char recv_msg[msg_count];
24        MPI_Status status;
25        for (int i = 1; i < num_procs; i++)
26        {
27            MPI_Recv(recv_msg, msg_count, MPI_CHAR, i, tag, MPI_COMM_WORLD, &status);
28            printf("rank id %d : %s\n", i, recv_msg);
29        }
30    }
31    else
32    {
33        char msg[msg_count] = {'H', 'e', 'l', 'l', 'o', ' ', ' ', 'F', 'r', 'o', 'm', ' ', ' ', char('0' + rank)};
34        MPI_Send(msg, msg_count, MPI_CHAR, 0, tag, MPI_COMM_WORLD);
35    }
36
37    MPI_Finalize();
38 }
```

The output window on the right shows the command prompt with the following commands and output:

```
PS D:\MNIT Projects\PDC Lab\Assignment 3> g++ .\mpi_functions.cpp -I "D:\Downloads\MPI\MS_MPI\SDK\Include" -L "D:\Downloads\MPI\MS_MPI\SDK\Lib\x64" -lmsmpi
PS D:\MNIT Projects\PDC Lab\Assignment 3> mpiexec.exe -n 10 .\a.exe
I am 2 of 10
I am 1 of 10
I am 3 of 10
I am 6 of 10
I am 5 of 10
I am 7 of 10
I am 4 of 10
I am 8 of 10
I am 9 of 10
I am 0 of 10
rank id 1 : Hello From 1
rank id 2 : Hello From 2
rank id 3 : Hello From 3
rank id 4 : Hello From 4
rank id 5 : Hello From 5
rank id 6 : Hello From 6
rank id 7 : Hello From 7
rank id 8 : Hello From 8
rank id 9 : Hello From 9
PS D:\MNIT Projects\PDC Lab\Assignment 3>
```

**Int MPI\_Init (int \*argc, char \*\*argv) :** Initialize MPI.

**Int MPI\_Finalize() :** Exit MPI.

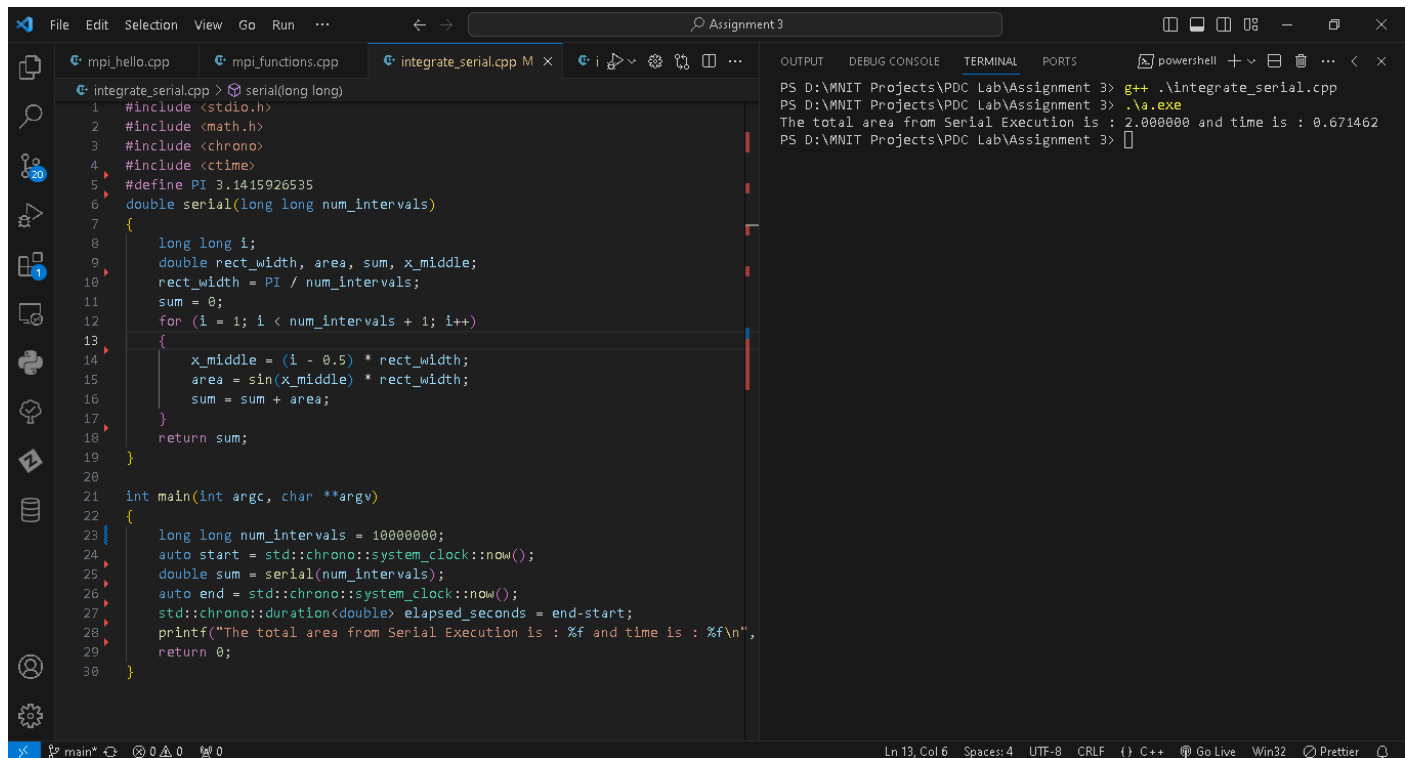
**Int MPI\_Comm\_size(MPI\_Comm comm, int \*size) :** Determine number of the processors within a communicator.

**Int MPI\_Comm\_rank(MPI\_Comm comm, int \*rank) :** Determine processor rank within a communicator.

**Int MPI\_Send(void \*buf, int count, MPI\_Datatype datatype, int dest, int tag, MPI\_Comm comm) :** Send a message to destination.

**Int MPI\_Recv(void \*buf, int count, MPI\_Datatype datatype, int source, int tag, MPI\_Comm comm, MPI\_Status \*status) :** Receive a message from source.

4. You need to use MPI to parallelize the serial program integrate.c (given below), which integrates function  $\sin(X)$  over the range from 0 to  $\pi$  using  $N$  intervals, where  $N$  is an argument of the program.
- Ans. Serial Program:**

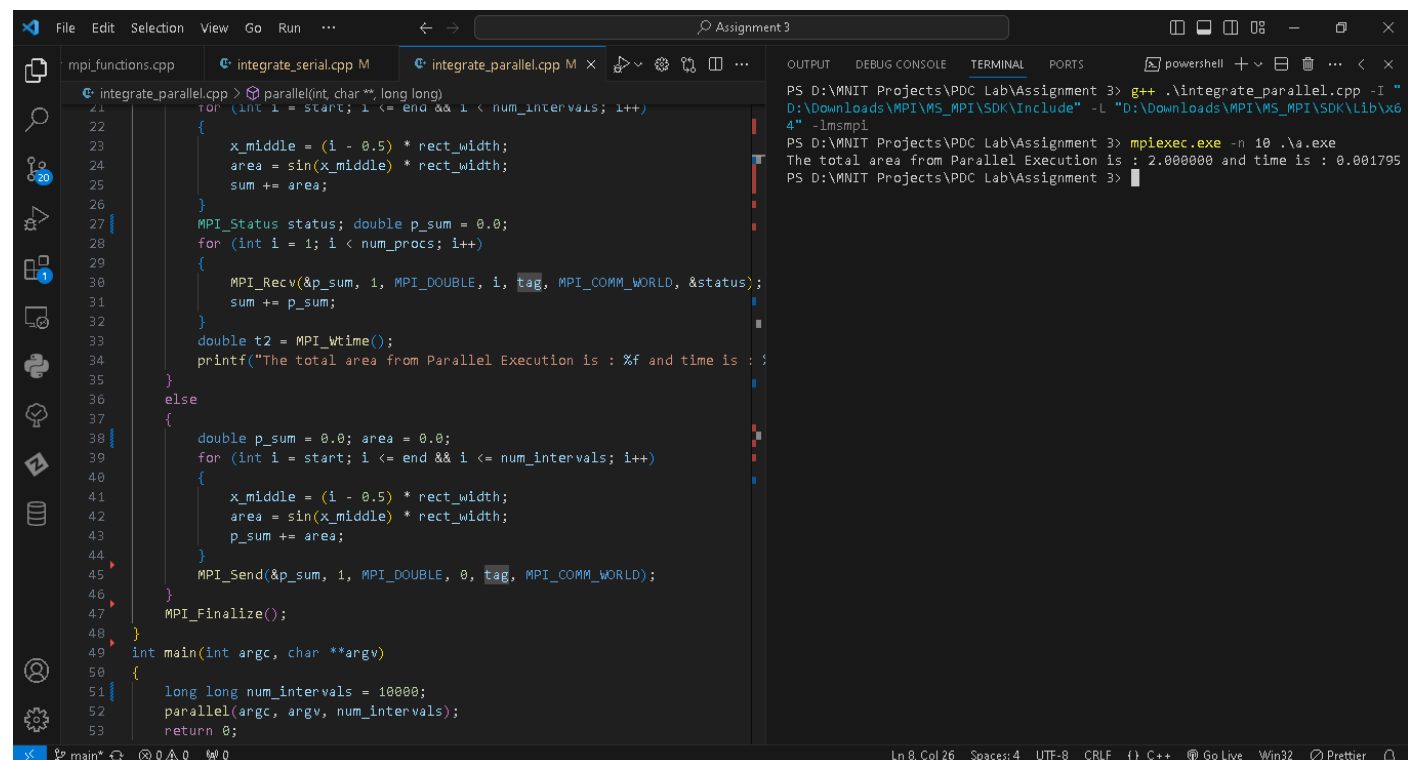


```
1 #include <stdio.h>
2 #include <math.h>
3 #include <chrono>
4 #include <ctime>
5 #define PI 3.1415926535
6 double serial(long long num_intervals)
7 {
8     long long i;
9     double rect_width, area, sum, x_middle;
10    rect_width = PI / num_intervals;
11    sum = 0;
12    for (i = 1; i < num_intervals + 1; i++)
13    {
14        x_middle = (i - 0.5) * rect_width;
15        area = sin(x_middle) * rect_width;
16        sum = sum + area;
17    }
18    return sum;
19 }
20
21 int main(int argc, char **argv)
22 {
23     long long num_intervals = 10000000;
24     auto start = std::chrono::system_clock::now();
25     double sum = serial(num_intervals);
26     auto end = std::chrono::system_clock::now();
27     std::chrono::duration<double> elapsed_seconds = end-start;
28     printf("The total area from Serial Execution is : %f and time is : %f\n",
29         sum, elapsed_seconds.count());
30     return 0;
31 }
```

OUTPUT

```
PS D:\MNIT Projects\PDC Lab\Assignment 3> g++ .\integrate_serial.cpp
PS D:\MNIT Projects\PDC Lab\Assignment 3> .\a.exe
The total area from Serial Execution is : 2.000000 and time is : 0.671462
PS D:\MNIT Projects\PDC Lab\Assignment 3>
```

## Parallel Program:



```
1 #include <stdio.h>
2 #include <math.h>
3 #include <chrono>
4 #include <ctime>
5 #define PI 3.1415926535
6 double serial(long long num_intervals)
7 {
8     long long i;
9     double rect_width, area, sum, x_middle;
10    rect_width = PI / num_intervals;
11    sum = 0;
12    for (i = 1; i < num_intervals + 1; i++)
13    {
14        x_middle = (i - 0.5) * rect_width;
15        area = sin(x_middle) * rect_width;
16        sum = sum + area;
17    }
18    return sum;
19 }
20
21 int main(int argc, char **argv)
22 {
23     long long num_intervals = 10000;
24     parallel(argc, argv, num_intervals);
25     return 0;
26 }
```

OUTPUT

```
PS D:\MNIT Projects\PDC Lab\Assignment 3> g++ .\integrate_parallel.cpp -I "D:\Downloads\MPI\MS_MPI\SDK\Include" -L "D:\Downloads\MPI\MS_MPI\SDK\Lib\x64" -lmmpi
PS D:\MNIT Projects\PDC Lab\Assignment 3> mpiexec.exe -n 10 .\a.exe
The total area from Parallel Execution is : 2.000000 and time is : 0.001795
PS D:\MNIT Projects\PDC Lab\Assignment 3>
```