Name : Ahmed Mohamed Aboelfetouh Mahmoud

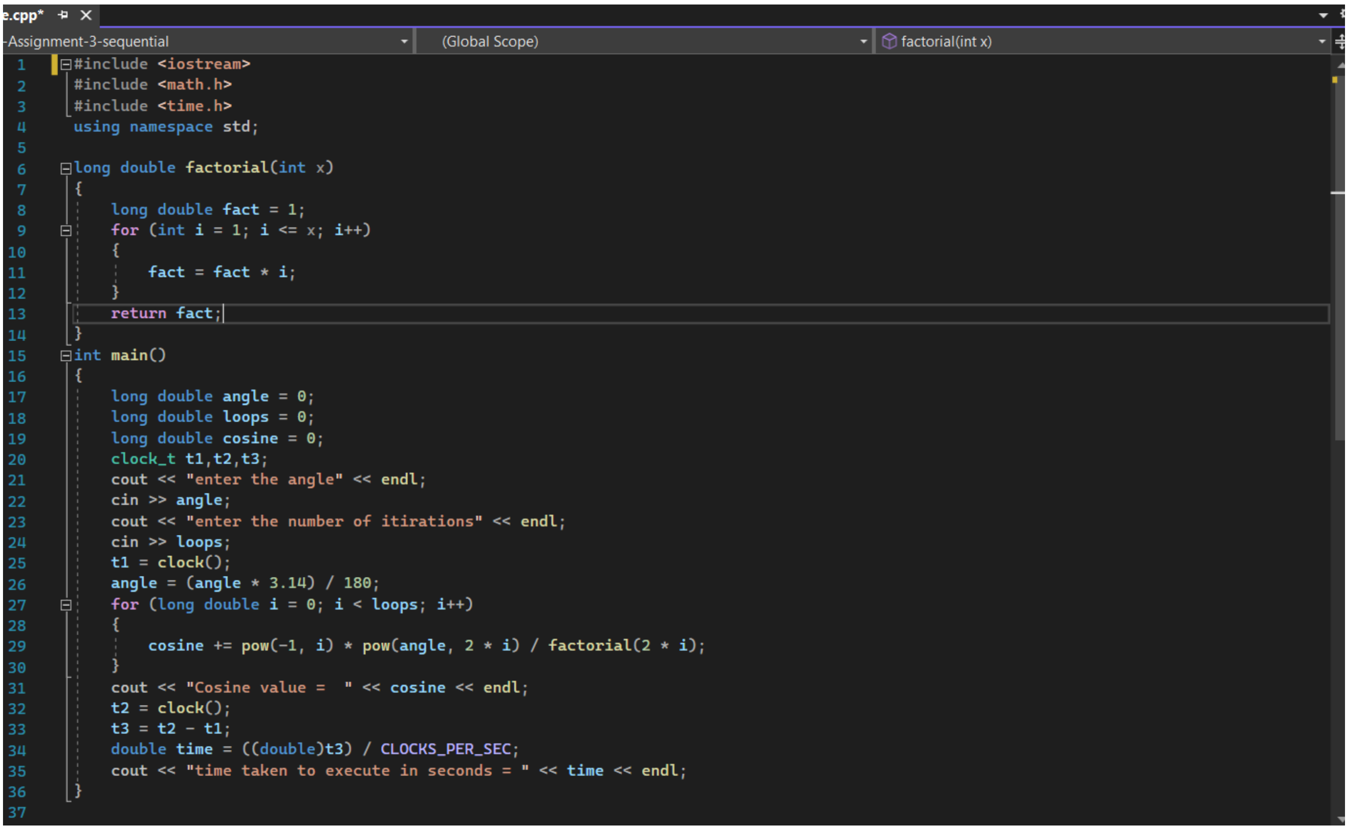
Sec: 1

Code : 1700154

Distributed Computer Systems

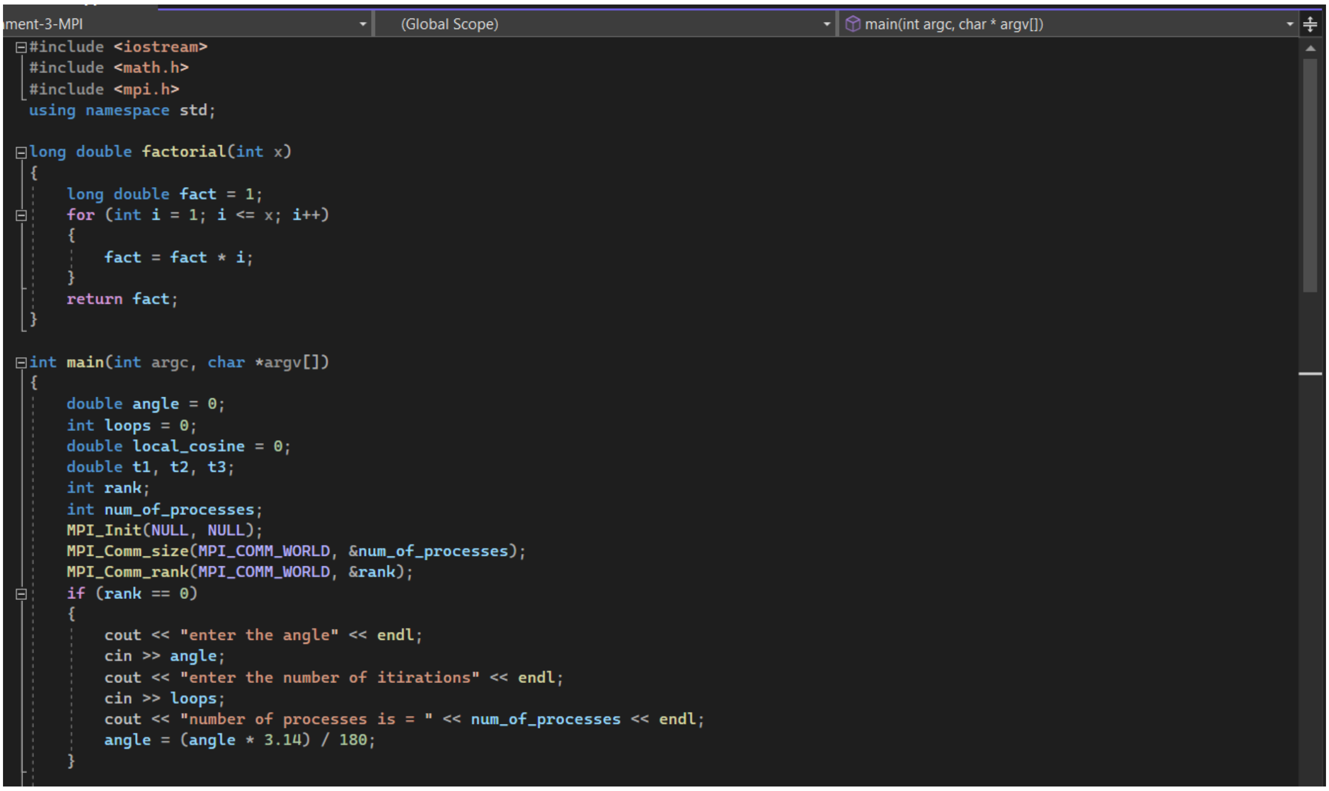
Assignment 3

1. Sequential code:

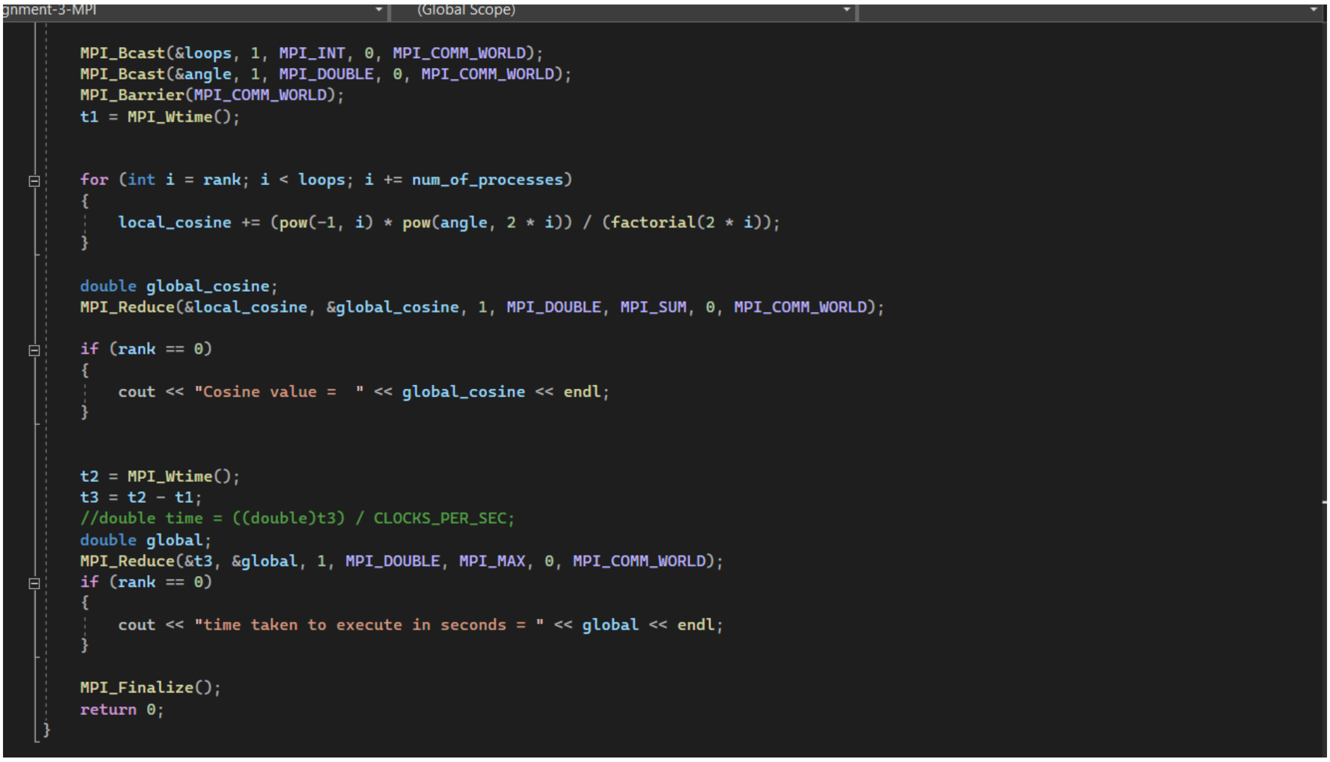


```cpp
#include <iostream>
#include <math.h>
#include <time.h>
using namespace std;

long double factorial(int x)
{
    long double fact = 1;
    for (int i = 1; i <= x; i++)
    {
        fact = fact * i;
    }
    return fact;
}
int main()
{
    long double angle = 0;
    long double loops = 0;
    long double cosine = 0;
    clock_t t1,t2,t3;
    cout << "enter the angle" << endl;
    cin >> angle;
    cout << "enter the number of itirations" << endl;
    cin >> loops;
    t1 = clock();
    angle = (angle * 3.14) / 180;
    for (long double i = 0; i < loops; i++)
    {
        cosine += pow(-1, i) * pow(angle, 2 * i) / factorial(2 * i);
    }
    cout << "Cosine value =  " << cosine << endl;
    t2 = clock();
    t3 = t2 - t1;
    double time = ((double)t3) / CLOCKS_PER_SEC;
    cout << "time taken to execute in seconds = " << time << endl;
}
```

2. MPI-code:



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

long double factorial(int x)
{
    long double fact = 1;
    for (int i = 1; i <= x; i++)
    {
        fact = fact * i;
    }
    return fact;
}

int main(int argc, char *argv[])
{
    double angle = 0;
    int loops = 0;
    double local_cosine = 0;
    double t1, t2, t3;
    int rank;
    int num_of_processes;
    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &num_of_processes);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (rank == 0)
    {
        cout << "enter the angle" << endl;
        cin >> angle;
        cout << "enter the number of itirations" << endl;
        cin >> loops;
        cout << "number of processes is = " << num_of_processes << endl;
        angle = (angle * 3.14) / 180;
    }
```



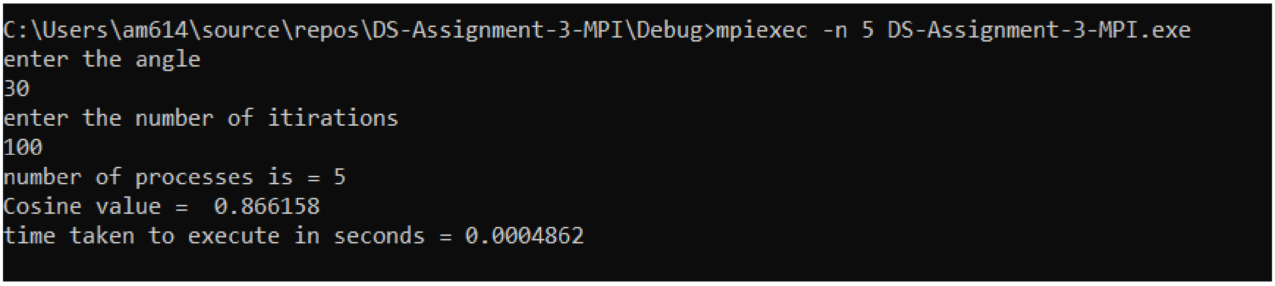
```
    MPI_Bcast(&loops, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&angle, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    MPI_Barrier(MPI_COMM_WORLD);
    t1 = MPI_Wtime();


    for (int i = rank; i < loops; i += num_of_processes)
    {
        local_cosine += (pow(-1, i) * pow(angle, 2 * i)) / (factorial(2 * i));
    }

    double global_cosine;
    MPI_Reduce(&local_cosine, &global_cosine, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {
        cout << "Cosine value =  " << global_cosine << endl;
    }


    t2 = MPI_Wtime();
    t3 = t2 - t1;
    //double time = ((double)t3) / CLOCKS_PER_SEC;
    double global;
    MPI_Reduce(&t3, &global, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
    if (rank == 0)
    {
        cout << "time taken to execute in seconds = " << global << endl;
    }

    MPI_Finalize();
    return 0;
}
```

MPI functions used in the code:

- MPI_Init(NULL, NULL);
  Used for Initialization of the MPI environment
- MPI_Comm_size(MPI_COMM_WORLD, &num_of_processes);
  Used to Get the number of processes
- int rank;
- MPI_Comm_rank(MPI_COMM_WORLD, &rank);
- Used to Get the rank of the process
- Here the process of rank 0 is the process responsible to communicate with the user
- MPI_Bcast(&loops, 1, MPI_INT, 0, MPI_COMM_WORLD);
  Used to Broadcast the iterations to the rest of processes
- MPI_Bcast(&angle, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
  Used to Broadcast the angle to the rest of processes
- MPI_Barrier(MPI_COMM_WORLD);
  (No process go beyond this line until all of then reaches it)
- double global_pi;
  MPI_Reduce(&local_cosine, &global_cosine, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
  Reduce operation to sum all local PIs into one variable in the root process (p0)
- MPI_Reduce(&t3, &global, 1, MPI_DOUBLE, MPI_MAX, 0, MPI_COMM_WORLD);
  Get the max time value from all processes to print

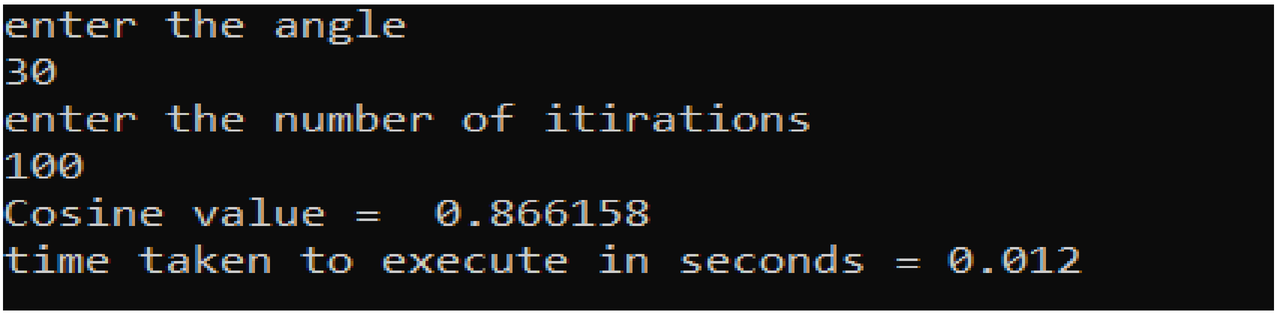
The different between time of sequential and 5 processes MPI:

1. 5 processes MPI:



```
C:\Users\am614\source\repos\DS-Assignment-3-MPI\Debug>mpiexec -n 5 DS-Assignment-3-MPI.exe
enter the angle
30
enter the number of itirations
100
number of processes is = 5
Cosine value =  0.866158
time taken to execute in seconds = 0.0004862
```

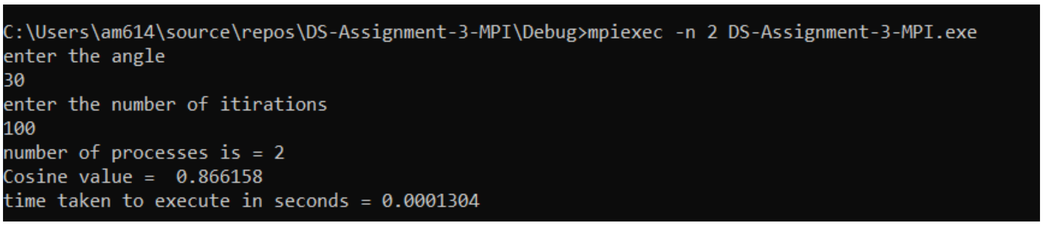
2. Sequential:



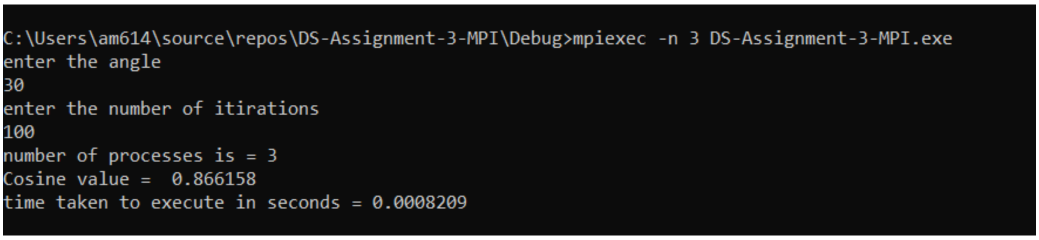
```
enter the angle
30
enter the number of itirations
100
Cosine value =  0.866158
time taken to execute in seconds = 0.012
```

- The time of the MPI is much smaller than the time of the sequential because of the 5 processes we created in the MPI example is working in parallel so they took much less time than the sequential method.

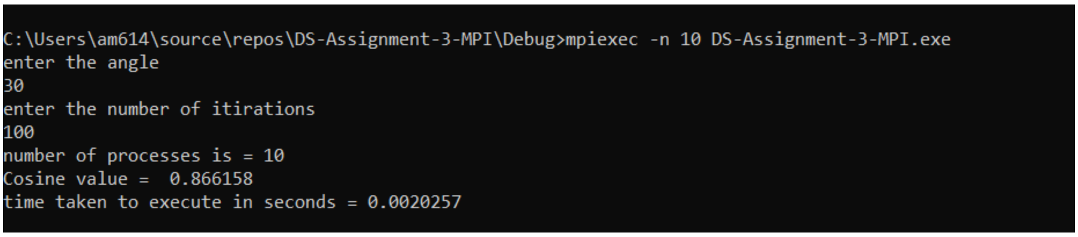
Time of the MPI code with different number of processes:



```
C:\Users\am614\source\repos\DS-Assignment-3-MPI\Debug>mpiexec -n 2 DS-Assignment-3-MPI.exe
enter the angle
30
enter the number of itirations
100
number of processes is = 2
Cosine value =  0.866158
time taken to execute in seconds = 0.0001304
```



```
C:\Users\am614\source\repos\DS-Assignment-3-MPI\Debug>mpiexec -n 3 DS-Assignment-3-MPI.exe
enter the angle
30
enter the number of itirations
100
number of processes is = 3
Cosine value =  0.866158
time taken to execute in seconds = 0.0008209
```



```
C:\Users\am614\source\repos\DS-Assignment-3-MPI\Debug>mpiexec -n 10 DS-Assignment-3-MPI.exe
enter the angle
30
enter the number of itirations
100
number of processes is = 10
Cosine value =  0.866158
time taken to execute in seconds = 0.0020257
```