Assignment 2

Student: Ubaid ur Rehman
Lecturer: Sir Ali Sayyad

This assignment is about PDC codes in which Open MPI is configured.

Task 1

- Workload distribution is mostly equal among all processors. However, the master does a bit more work as it first distributes chunks and then computes its own.
- No, this design does not scale well for large arrays due to communication and scalability issues while sending and receiving data chunks to other processors.

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_1.c -o t1
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t1
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

Figure 1: Serial code for iterative printing.

Task 2

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_2.c -o t2
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t2
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

Figure 2: Parallel code for addition of arrays.

- The need for ‘Waitall()‘ arises because ‘MPI_Isend()‘ and ‘MPI_Irecv()‘ are non-blocking. Unlike ‘MPI_Send()‘ and ‘MPI_Recv()‘, they do not wait for the communication to complete. So, a wait barrier like ‘MPI_Waitall()‘ is required to ensure all communications finish before proceeding.
- Without proper waiting, the program can crash or behave unexpectedly. For example, if a receive call is made before the corresponding send completes, it may lead to runtime errors.

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t2
1 4 9 16 5 6 7 8 9 10 11 12 13 14 15 16
[dr-pc-HP-EliteBook-840-G6:95933] *** Process received signal ***
[dr-pc-HP-EliteBook-840-G6:95933] Signal: Segmentation fault (11)
[dr-pc-HP-EliteBook-840-G6:95933] Signal code:  (128)
[dr-pc-HP-EliteBook-840-G6:95933] Failing at address: (nil)
--------------------------------------------------------------------------
Primary job  terminated normally, but 1 process returned
a non-zero exit code. Per user-direction, the job has been aborted.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun noticed that process rank 0 with PID 0 on node dr-pc-HP-EliteBook-840-G6 exited on signal 11 (Segmentation fault).
--------------------------------------------------------------------------
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

Figure 3: Runtime error caused by missing wait after non-blocking send.

Task 3

- Tags provide uniqueness in MPI messages. Since processes may send and receive simultaneously, tags help identify and separate different messages.
- Without tags, a receiver cannot distinguish which message corresponds to which data, leading to confusion and incorrect results.

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_3.c -o t3
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t3
Final Results:
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256 1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

Figure 4: Use of tags to differentiate MPI messages.

Task 4

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_4.c -o t4
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t4
Final Count: 16
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

- The main pitfall normally faced in ring algorithm is if one node fail then whole ring fail.
- In a bi-directional ring, each process communicates with both neighbors, enabling faster data transfer and improved fault tolerance. However, it introduces added complexity in synchronization and handling potential message collisions.

Task 5

```
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_5_a.c -o t5
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t5
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256
Execution time: 0.00016 seconds
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

```
Execution time: 0.00016 seconds
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpicc task_5_b.c -o t6
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$ mpirun -np 4 ./t6
1 4 9 16 25 36 49 64 81 100 121 144 169 196 225 256
Execution time: 0.00005 seconds
(base) dr-pc@dr-pc-HP-EliteBook-840-G6:~/Desktop$
```

Figure 5: Using `#pragma omp atomic`.

-
