Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

Year: 2021-22 **Semester:** 1

Course: High Performance Computing Lab

Practical No. 5

Exam Seat No:

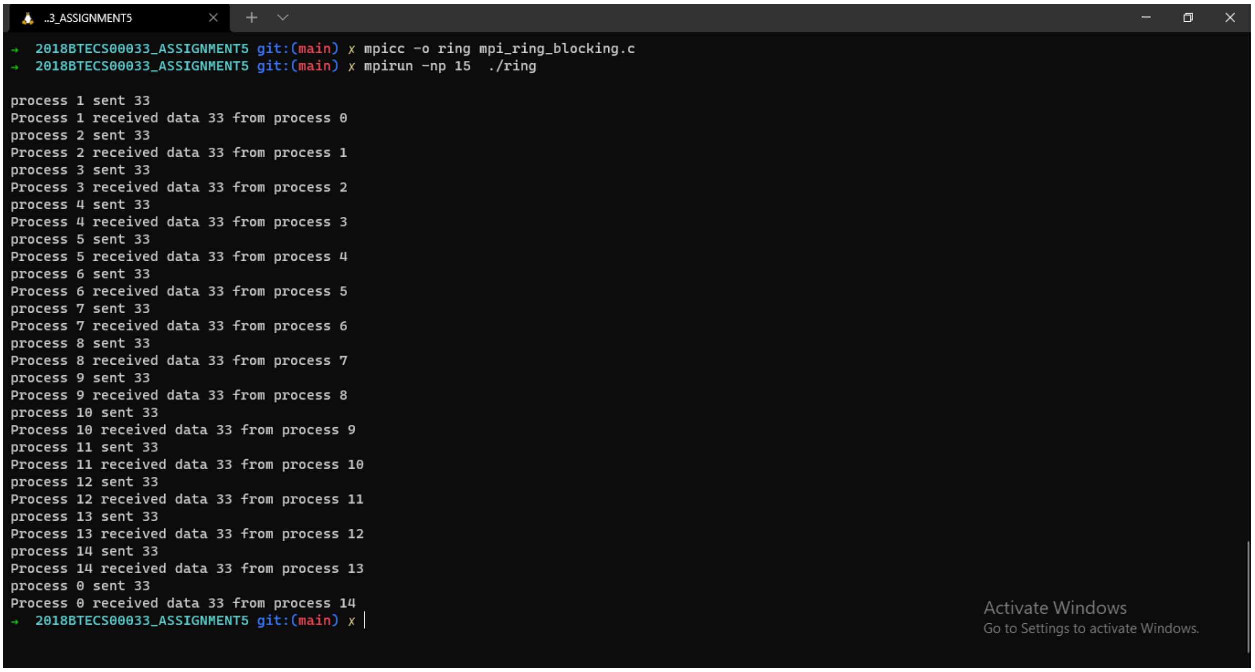
1. 2018BTECS00033 - Mahendra Bhimrao Gharge

Problem Statement 1:
Implement blocking and non-blocking MPI send & receive to demonstrate Nearest neighbour exchange of data in a ring topology.

Code:

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

int main(int argc, char **argv)
{
    MPI_Init(NULL, NULL);
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int size;
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int data = 33;
    if (rank != 0)
    {
        MPI_Recv(&data, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        printf("Process %d received data %d from process %d\n", rank,
data, rank - 1);
    }
    else
    {
    }
    MPI_Send(&data, 1, MPI_INT, (rank + 1) % size, 0, MPI_COMM_WORLD);
    printf("process %d sent %d \n", (rank + 1) % size, data);
    if (rank == 0)
    {
        MPI_Recv(&data, 1, MPI_INT, size - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        printf("Process %d received data %d from process %d\n", rank,
data, size - 1);
    }
    MPI_Finalize();
}
```

Screenshot 1:



```
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpicc -o ring mpi_ring_blocking.c
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpirun -np 15  ./ring

process 1 sent 33
Process 1 received data 33 from process 0
process 2 sent 33
Process 2 received data 33 from process 1
process 3 sent 33
Process 3 received data 33 from process 2
process 4 sent 33
Process 4 received data 33 from process 3
process 5 sent 33
Process 5 received data 33 from process 4
process 6 sent 33
Process 6 received data 33 from process 5
process 7 sent 33
Process 7 received data 33 from process 6
process 8 sent 33
Process 8 received data 33 from process 7
process 9 sent 33
Process 9 received data 33 from process 8
process 10 sent 33
Process 10 received data 33 from process 9
process 11 sent 33
Process 11 received data 33 from process 10
process 12 sent 33
Process 12 received data 33 from process 11
process 13 sent 33
Process 13 received data 33 from process 12
process 14 sent 33
Process 14 received data 33 from process 13
process 0 sent 33
Process 0 received data 33 from process 14
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ |
```

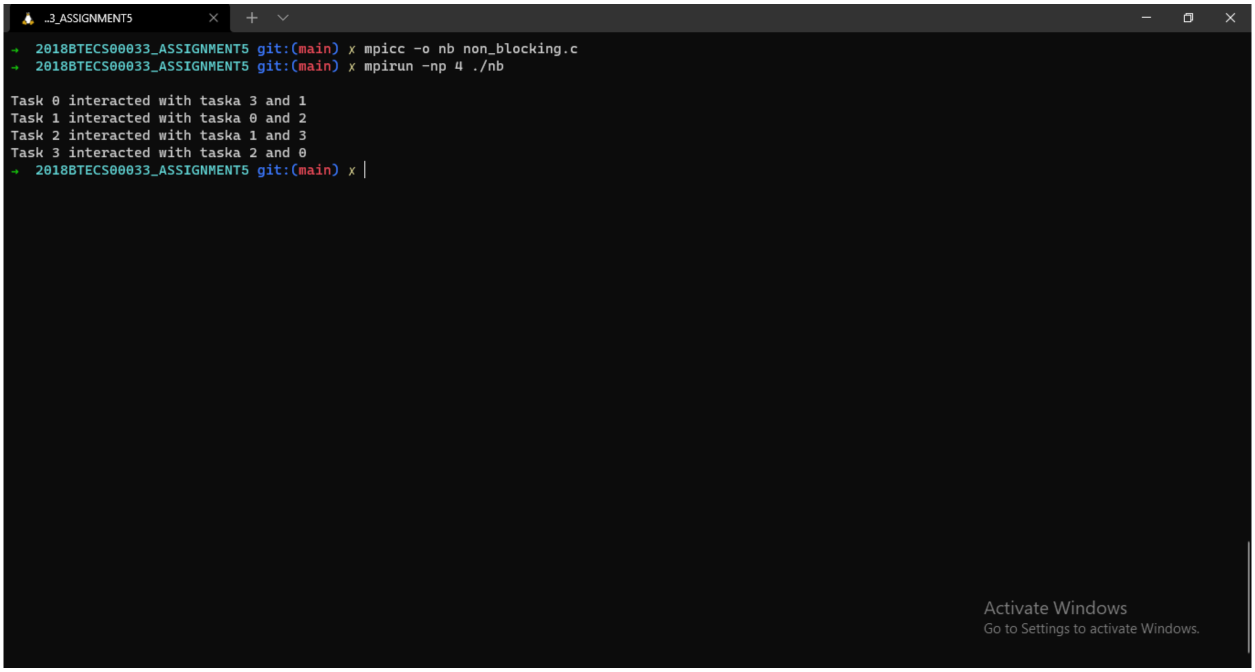
Information 1: Ring topology blocking send and receive with 15 processes.

Problem Statement 2:

IImplement a MPI program to give an example of non-blocking send and receive between four processes.

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

int main()
{
    int numtasks, rank, prev, next, sendVal = 33, recVal = 19, tag1 = 0;

    MPI_Request reqs[4];
    MPI_Status stats[4];

    MPI_Init(NULL, NULL);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    prev = rank - 1;
    next = rank + 1;

    if (rank == 0)
        prev = numtasks - 1;
    if (rank == numtasks - 1)
        next = 0;

    MPI_Irecv(&recVal, 1, MPI_INT, prev, tag1, MPI_COMM_WORLD, &reqs[0]);

    MPI_Isend(&sendVal, 1, MPI_INT, next, tag1, MPI_COMM_WORLD, &reqs[1]);

    PMPI_Waitall(2, reqs, stats);
    printf("Task %d interacted with taska %d and %d \n", rank, prev,
next);

    MPI_Finalize();
    return 0;
}
```

Screenshot 2:



```
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpicc -o nb non_blocking.c
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpirun -np 4 ./nb

Task 0 interacted with taska 3 and 1
Task 1 interacted with taska 0 and 2
Task 2 interacted with taska 1 and 3
Task 3 interacted with taska 2 and 0
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ |
```

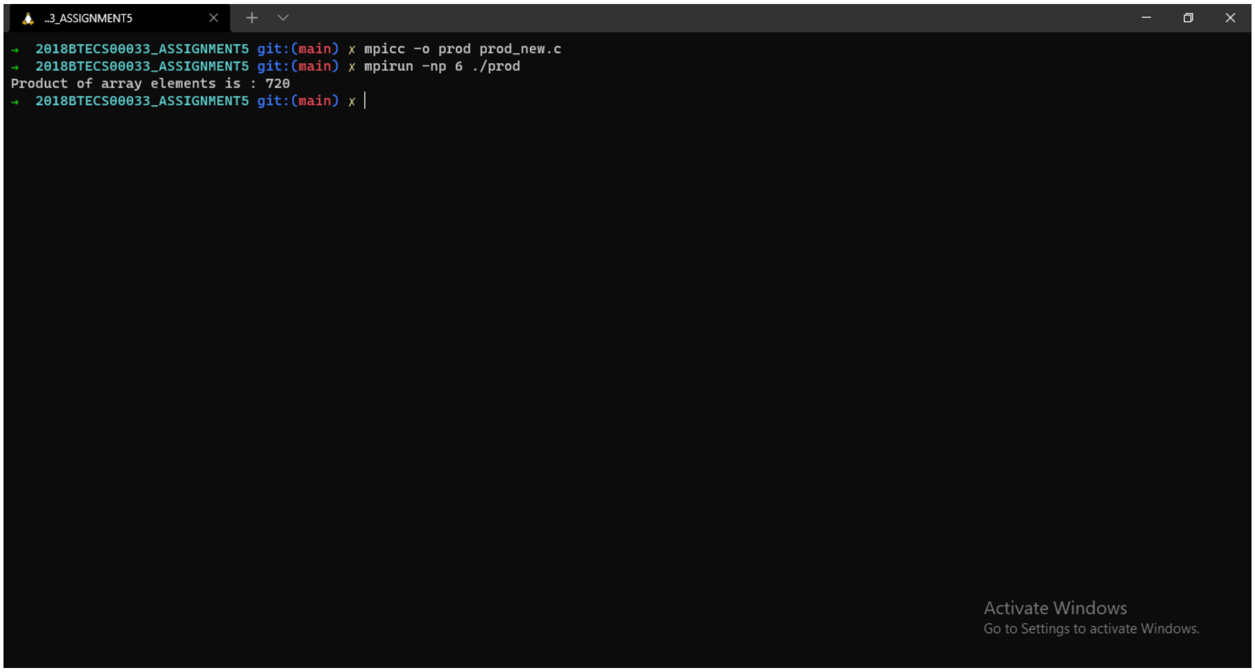
Information 2: Non-blocking send and receive between 4 processes.

Problem Statement 3:

Write a MPI program to find the product of all the elements of an array A of size n using m number of processes. The two sums then are added to get the final result.

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

int main()
{
    int arr[6] = {1, 2, 3, 4, 5, 6};
    MPI_Init(NULL, NULL);
    int size, rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int prod = 1;

    prod = arr[rank];

    int prod_main;
    MPI_Reduce(&prod, &prod_main, 1, MPI_INT, MPI_PROD, 0,
MPI_COMM_WORLD);

    if (rank == 0)
        printf("Product of array elements is : %d\n", prod_main);

    MPI_Finalize();

    return 0;
}
```

Screenshot 3:



```
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpicc -o prod prod_new.c
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ mpirun -np 6 ./prod
Product of array elements is : 720
→  2018BTECS00033_ASSIGNMENT5 git:(main) ✗ |
```

Information 3: Product of array elements using MPI

Github Link: https://github.com/g-mahendra/HPC_LAB_ASSIGNMENTS