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:2018BTECS00100**

1. Exam Seat Number – Prakash Singh

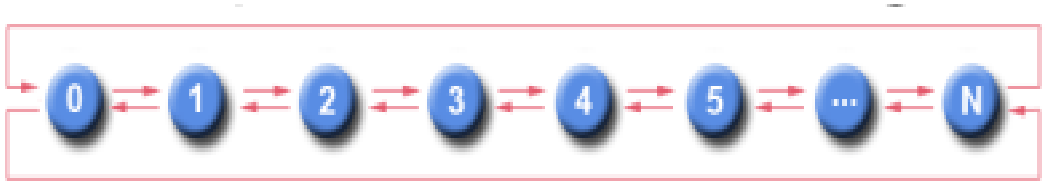
**Problem Statement 1:**Implement blocking and non-blocking MPI send & receive to demonstrate Nearest neighbour exchange of data in a ring topology.

**Screenshot #1:**

```
prax@prakx-ideapad:~/Desktop/HPC/5$ mpicc -o  1 1.c
prax@prakx-ideapad:~/Desktop/HPC/5$ mpiexec -np 4 ./1
Process 2 received with tag 1 from process 3
Process 0 received with tag 2 from process 3
Process 3 received with tag 1 from process 0
Process 1 received with tag 2 from process 0
Process 0 received with tag 1 from process 1
Process 2 received with tag 2 from process 1
Process 1 received with tag 1 from process 2
Process 3 received with tag 2 from process 2
prax@prakx-ideapad:~/Desktop/HPC/5$
```

**Information #:**

Implemented blocking and non-blocking MPI send & receive to demonstrate Nearest neighbour exchange of data in a ring topology.



**Problem Statement 2:** Implement a MPI program to give an example of non-blocking send and receive between four processes.

**Screenshot #1:**

```
prax@prakx-ideapad:~/Desktop/HPC/5$ mpicc -o  2 2.c
prax@prakx-ideapad:~/Desktop/HPC/5$ mpiexec -np 4 ./2
Process 3 sent data 100 to  process 2
Process 3 received data 100 from process 0
Process 1 sent data 100 to  process 0
Process 1 received data 100 from process 2
Process 0 received data 100 from process 1
Process 2 sent data 100 to  process 1
Process 2 received data 100 from process 3
```

**Information :**

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

```c
#include <stdio.h>
#include <mpi.h>
int main(int argc, char** argv)
{
    int my_rank, nprocs;
    MPI_Request request;
    MPI_Status status;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
    int data = 100;

    if (my_rank!=0){

        MPI_Isend(&data, 1, MPI_INT, my_rank-1, 1, MPI_COMM_WORLD,&request);
        printf("Process %d sent data %d to  process %d\n", my_rank,data, my_rank-1);
        MPI_Wait(&request, &status);

    }
    if (my_rank!=nprocs){

        MPI_Irecv(&data, 1, MPI_INT, (my_rank+1)%nprocs , 0, MPI_COMM_WORLD, &request);
        printf("Process %d received data %d from process %d\n",my_rank,data,(my_rank+1)%nprocs);

    }



        MPI_Finalize();
}
```

**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.

**Screenshot 1:**

```
prax@prakx-ideapad:~/Desktop/HPC/5$ mpicc -o  3 3.c
prax@prakx-ideapad:~/Desktop/HPC/5$ mpiexec -np 2 ./3
Product  of array is : 1680
```

**Information :**

A MPI program to find the product of all the elements of an array A of size 6 using 2 number of processes. The two sums then are added to get the final result by using MPI_REDUCE.

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

int main()
{
    int prod_arr[6] = {1,2,4,5,6,7};
    MPI_Init(NULL, NULL);
    int size_world, my_rank;
    size_t n = sizeof(prod_arr)/sizeof(prod_arr[0]);

    //initilaizing product
    int product = 1;
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size_world);


int partial_prod = 1;
    if (my_rank==0){

        for (int i = 0;i<(my_rank+(n/size_world));i++){
            partial_prod *=prod_arr[i];
        }
    }
    else{

        for (int i = 3 ;i<n;i++){
            partial_prod *=prod_arr[i];
        }
    }


    MPI_Reduce(&partial_prod, &product, 1, MPI_INT, MPI_PROD, 0, MPI_COMM_WORLD);

     if (my_rank == 0)
 {
    printf("Product  of array is : %d\n", product);
 }

    MPI_Finalize();

    return 0;
}
```

**Github Link: https://github.com/prakx1/HPC-LAB/tree/master/5**