**Class:** Final Year (Computer Science and Engineering)

**Year:** 2023-24 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 7**

**Name:** Sumit Narake

**PRN:**2020BTECS00023

**Batch :** B2

**Information:**

**MPI\_Init:**

Description: Initializes the MPI environment. It must be called before any other MPI function.

Syntax: int MPI\_Init(int\* argc, char\*\*\* argv);

**MPI\_Finalize:**

Description: Finalizes the MPI environment. It should be called when you're done with MPI operations.

Syntax: int MPI\_Finalize(void);

**MPI\_Comm\_rank:**

Description: Retrieves the rank of the calling process within a communicator.

Syntax: int MPI\_Comm\_rank(MPI\_Comm comm, int\* rank);

**MPI\_Comm\_size:**

Description: Retrieves the total number of processes in a communicator.

Syntax: int MPI\_Comm\_size(MPI\_Comm comm, int\* size);

**MPI\_Send:**

Description: Sends data from the calling process to another process.

Syntax: int MPI\_Send(void\* buf, int count, MPI\_Datatype datatype, int dest, int tag, MPI\_Comm comm);

**MPI\_Recv:**

Description: Receives data into the calling process from another process.

Syntax: int MPI\_Recv(void\* buf, int count, MPI\_Datatype datatype, int source, int tag, MPI\_Comm comm, MPI\_Status\* status);

**Problem Statement 1:**

Implement a simple hello world program by setting number of processes equal to 10

Program:

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[]) {

    int rank, size;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if (size != 10) {

        printf("This program is intended to be run with exactly 10 processes.\n");

    } else {

        printf("Hello, world! I'm process %d of %d.\n", rank, size);

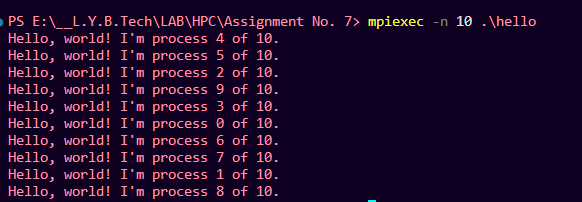
    }

    MPI\_Finalize();

    return 0;

}

**Screenshots:**

****

**Problem Statement 2:**

Implement a program to display rank and communicator group of five processes

**Program:**

#include <stdio.h>

#include "mpi.h"

int main(int argc, char \*argv[]) {

    int rank, size;

    MPI\_Comm new\_comm;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if (size != 5) {

        printf("Rank: %d of %d processes in the communicator group.\n", rank, size);

        printf("This program is intended to be run with exactly 5 processes but you don't have sufficient processors.\n");

    } else {

        MPI\_Comm\_dup(MPI\_COMM\_WORLD, &new\_comm);

        printf("Rank: %d of %d processes in the communicator group.\n", rank, size);

        MPI\_Comm\_free(&new\_comm);

    }

    MPI\_Finalize();

    return 0;

}

**Screenshots:**

