High Performance Computing Lab

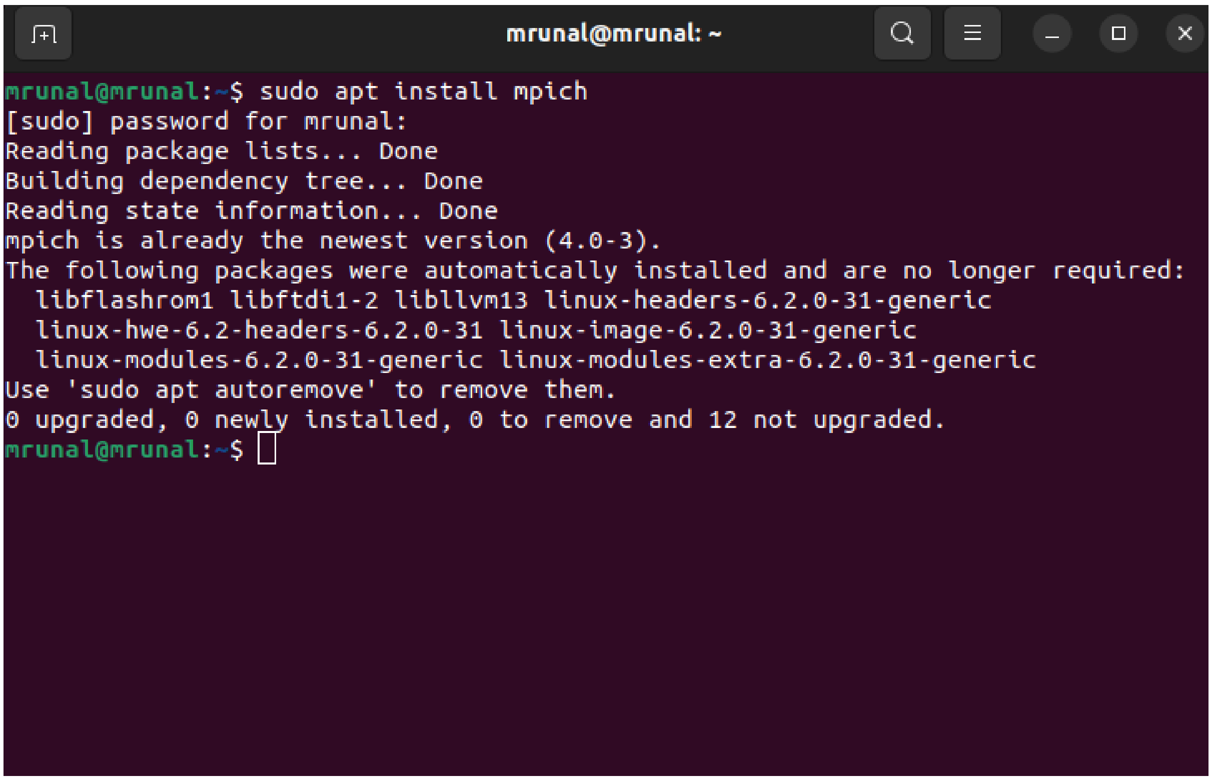
Practical No. 7

Name: Mrunal Anil Khade

PRN: 2020BTECS00057

Title of practical:

Installation of MPI & Implementation of basic functions of MPI



```
mrunal@mrunal:~$ sudo apt install mpich
[sudo] password for mrunal:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mpich is already the newest version (4.0-3).
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 libllvm13 linux-headers-6.2.0-31-generic
  linux-hwe-6.2-headers-6.2.0-31 linux-image-6.2.0-31-generic
  linux-modules-6.2.0-31-generic linux-modules-extra-6.2.0-31-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.
mrunal@mrunal:~$
```

Problem Statement 1:

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

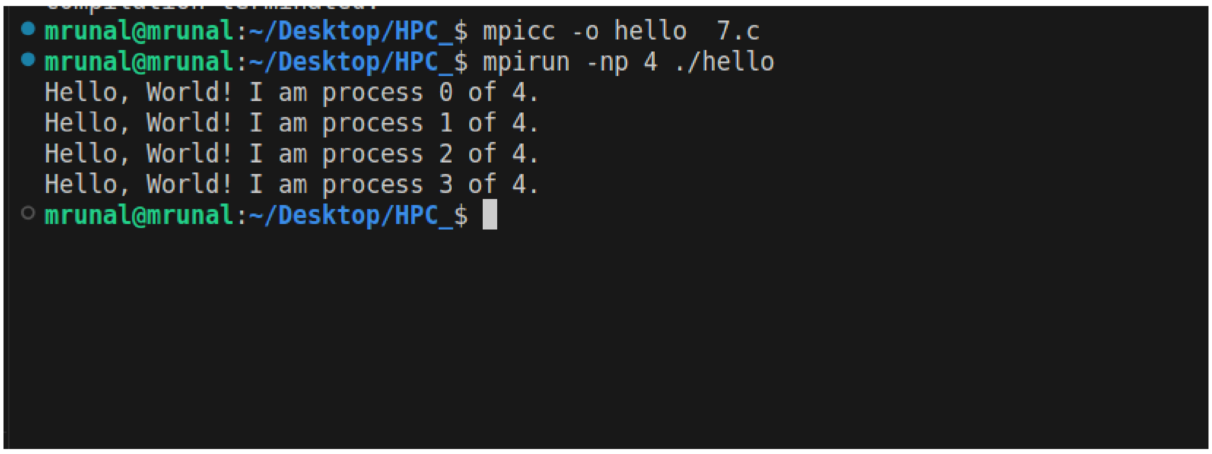
Screenshots:

Code:

```
//2020btecs00057


#include <mpi.h>
```

```
#include <stdio.h>

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

   MPI_Init(&argc, &argv);

   printf("Hello, world!\n");

   return 0;

}
```

Output:



```
● mrunal@mrunal:~/Desktop/HPC_$ mpicc -o hello  7.c
● mrunal@mrunal:~/Desktop/HPC_$ mpirun -np 4 ./hello
  Hello, World! I am process 0 of 4.
  Hello, World! I am process 1 of 4.
  Hello, World! I am process 2 of 4.
  Hello, World! I am process 3 of 4.
○ mrunal@mrunal:~/Desktop/HPC_$ 
```

Information 1:

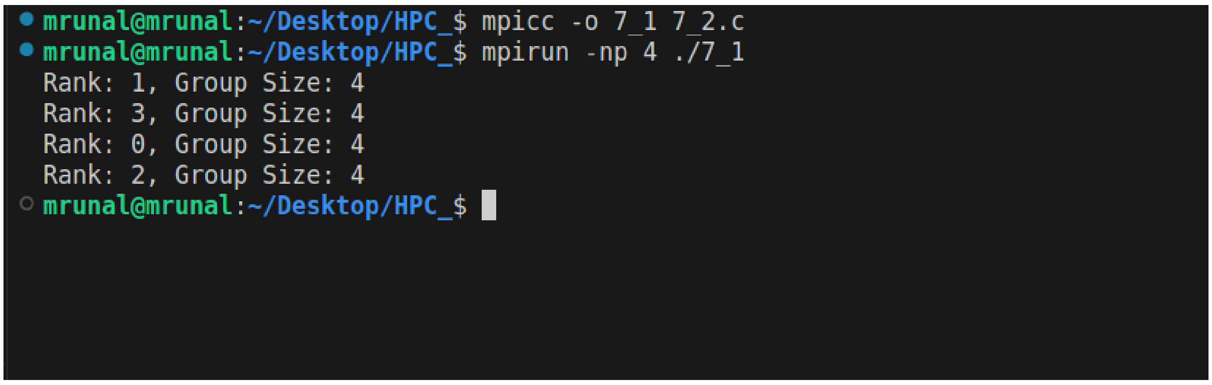
Problem Statement 2:

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

```
// 2020btecs00057



#include <mpi.h>

#include <stdio.h>
```

```c
int main(int argc, char *argv[]) {
   MPI_Init(&argc, &argv);


   int rank;
   MPI_Group group;


   MPI_Comm_group(MPI_COMM_WORLD, &group);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);


   int group_size;
   MPI_Group_size(group, &group_size);


   printf("Rank: %d, Group Size: %d\n", rank, group_size);


   MPI_Finalize();
   return 0;
}
```

Output:



```
● mrunal@mrunal:~/Desktop/HPC_$ mpicc -o 7_1 7_2.c
● mrunal@mrunal:~/Desktop/HPC_$ mpirun -np 4 ./7_1
  Rank: 1, Group Size: 4
  Rank: 3, Group Size: 4
  Rank: 0, Group Size: 4
  Rank: 2, Group Size: 4
○ mrunal@mrunal:~/Desktop/HPC_$ 
```

Information:

- The **main** function is the entry point of the program. It takes command-line arguments **argc** and **argv** which are typically used to pass arguments to the program.
- **MPI_Init(&argc, &argv)** is an MPI function that initializes the MPI environment. It is called at the beginning of every MPI program to set up communication between the MPI processes. The **argc** and **argv** arguments are usually passed to enable command-line options.
- **rank:** This variable will store the rank of the current MPI process.
- **group:** This variable will be used to represent the group of MPI processes.
- **MPI_Comm_group** is used to obtain the group associated with the communicator **MPI_COMM_WORLD. MPI_COMM_WORLD** is a predefined communicator that represents all processes. The group obtained is stored in the **group** variable.
- **MPI_Comm_rank** is used to retrieve the rank of the current process within the communicator **MPI_COMM_WORLD.** The rank is stored in the **rank** variable.
- Another integer variable, **group_size,** is declared to store the size (number of processes) in the group. **MPI_Group_size** is called with the **group** variable, and the size is stored in **group_size.**
- **MPI_Finalize()** is called to finalize the MPI environment. It should be called at the end of every MPI program to clean up resources and ensure proper termination of the MPI processes.