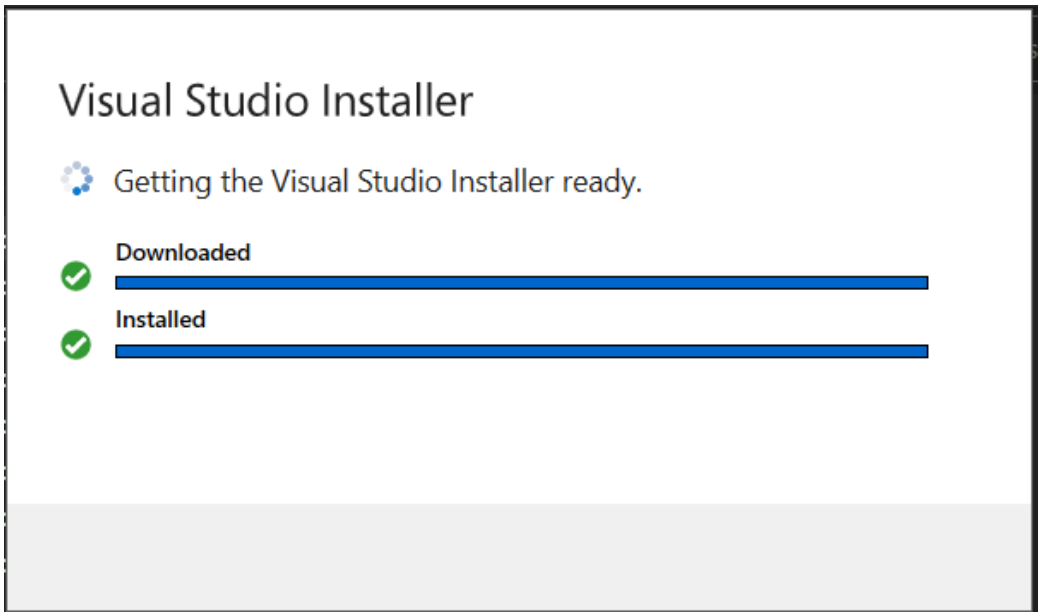
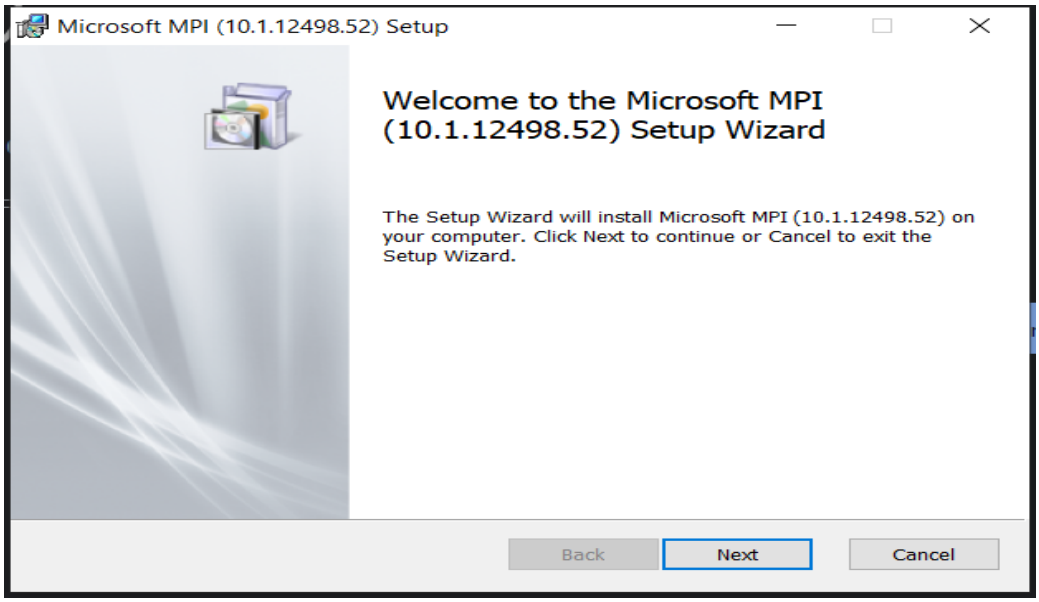
Assignment No. 7

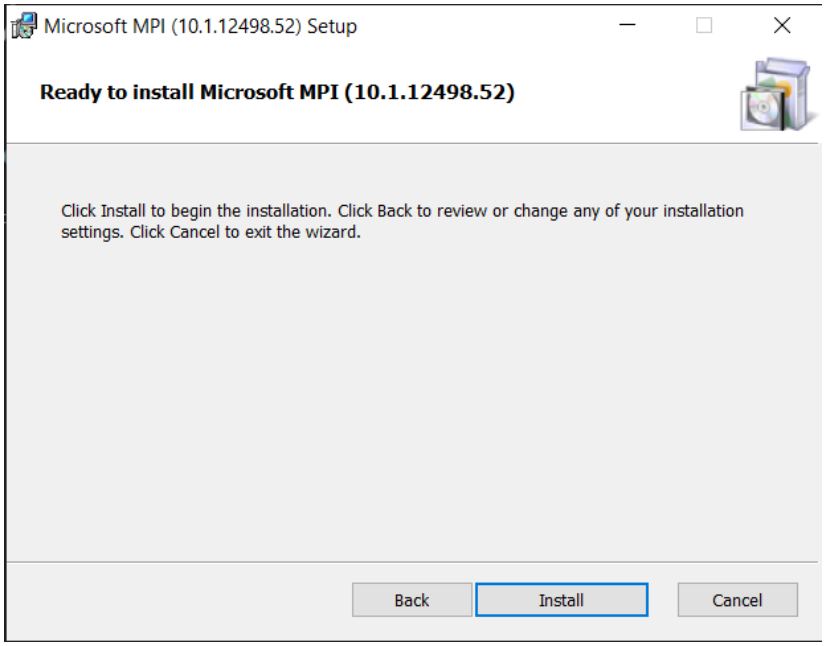
PRN: 2020BTECS00025
Name: Sahil Santosh Otari
Course: High Performance Computing Lab
Title of practical: Installation of MPI & Implementation of basic functions of MPI.

1. Installing Visual Studio

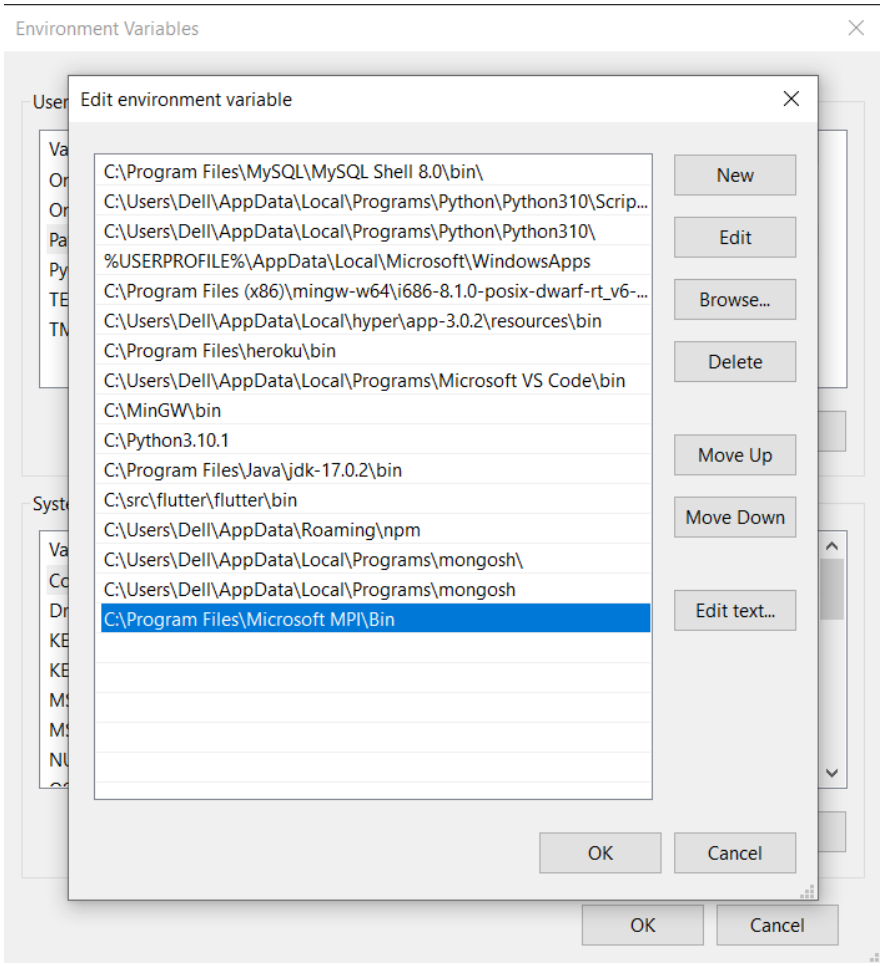


2. Download and install MPI for windows





3. Setting up path of MPI libraries.



4. Check if MPI Installed successfully.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.1348]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mpiexec -help
Microsoft MPI Startup Program [Version 10.1.12498.52]

Launches an application on multiple hosts.

Usage:

    mpiexec [options] executable [args] [ : [options] exe [args] : ... ]
    mpiexec -configfile <file name>

Common options:

-n <num_processes>
-env <env_var_name> <env_var_value>
-wdir <working_directory>
-hosts n host1 [m1] host2 [m2] ... hostn [mn]
-cores <num_cores_per_host>
-lines
-debug [0-3]
-logfile <log file>

Examples:

    mpiexec -n 4 pi.exe
    mpiexec -hosts 1 server1 master : -n 8 worker

For a complete list of options, run mpiexec -help2
For a list of environment variables, run mpiexec -help3

You can reach the Microsoft MPI team via email at askmpi@microsoft.com

C:\WINDOWS\system32>
```

Problem Statement 1:

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

Code:

```
#include <mpi.h>
#include <stdio.h>
int main(int argc, char* argv[]) {
    MPI_Init(&argc, &argv);
    int node;
    MPI_Comm_rank(MPI_COMM_WORLD, &node);

    printf("Hello World from Node %d\n", node);
    MPI_Finalize();
    return 0;
}
```

Output:

```
Windows PowerShell
PS C:\Users\Dell\source\repos\mpiProgram1\Debug> mpiexec mpiProgram1.exe
Hello World from Node 7
Hello World from Node 6
Hello World from Node 1
Hello World from Node 4
Hello World from Node 3
Hello World from Node 5
Hello World from Node 2
Hello World from Node 0
PS C:\Users\Dell\source\repos\mpiProgram1\Debug>
```

Information:

- MPI_Init(&argc,&argv);
  calls MPI_Init to initialize the MPI environment, and generally set up everything. This should be the first command executed in all programs. This routine takes pointers to argc and argv, looks at them, pulls out the purely MPI-relevant things, and generally fixes them so you can use command line arguments as normal.
- After doing everything else, the program calls MPI_Finalize, which generally terminates everything and shuts down MPI. This should be the last command executed in all programs.

Problem Statement 2:

Code:

```
#include <mpi.h>
#include <stdio.h>
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 is %d, Group Size is %d\n", rank, group_size);
    MPI_Finalize();
```

```
    return 0;
}
```

Output:

```
Windows PowerShell
PS C:\Users\Dell\source\repos\GroupSize\Debug> mpiexec GroupSize.exe
Rank is 4, Group Size is 8
Rank is 1, Group Size is 8
Rank is 6, Group Size is 8
Rank is 0, Group Size is 8
Rank is 7, Group Size is 8
Rank is 5, Group Size is 8
Rank is 2, Group Size is 8
Rank is 3, Group Size is 8
PS C:\Users\Dell\source\repos\GroupSize\Debug>
```

Information:

- MPI_Init(&argc,&argv);
  calls MPI_Init to initialize the MPI environment, and generally set up everything. This should be the first command executed in all programs. This routine takes pointers to argc and argv, looks at them, pulls out the purely MPI-relevant things, and generally fixes them so you can use command line arguments as normal.
- The MPI_Comm_group() function is used to create a group of processes within a communicator.
- The MPI_Comm_rank() is used to determine the rank of the calling process within a given communicator.
- The MPI_Group_size() to obtain the number of processes in a specified group.
- After doing everything else, the program calls MPI_Finalize, which generally terminates everything and shuts down MPI. This should be the last command executed in all programs.