Parallel and Distributed Computing
CSE4001
Fall Semester 2020-21

Lab Assignment 10

ISHAAN OHRI
18BCE0265

Aim:

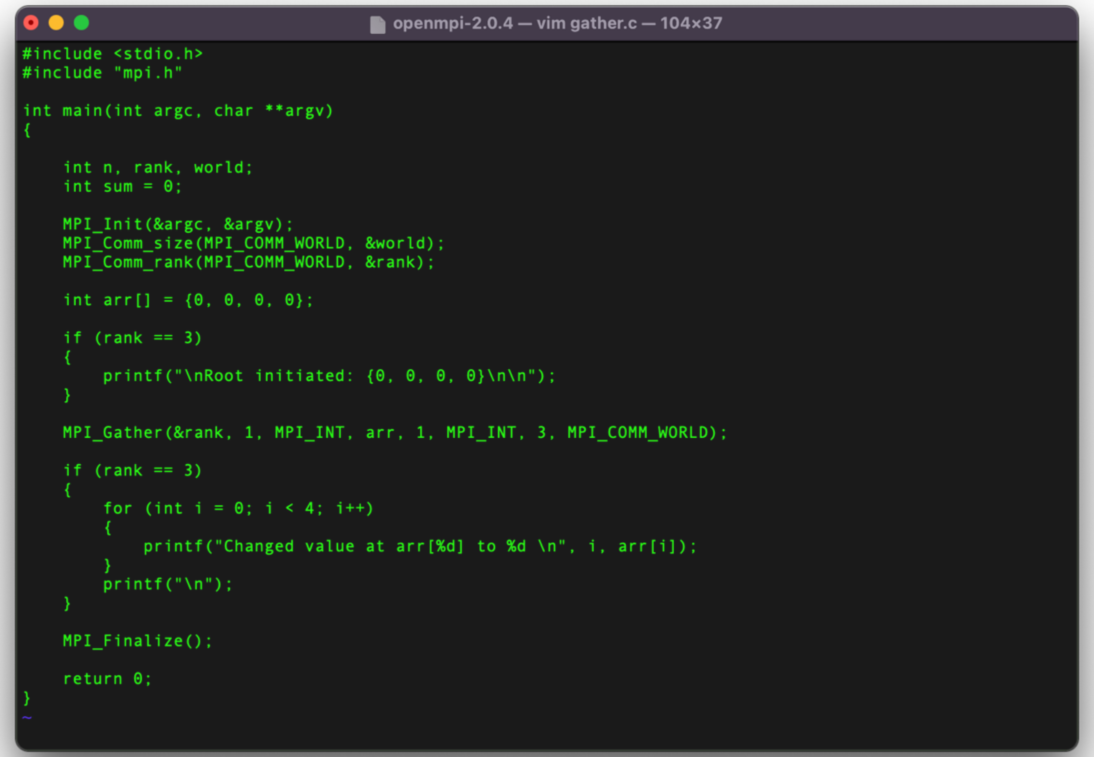
Assume the variable rank contains the process rank and root is 3. What will be stored in array b [ ] on each of four processes if each executes the following code fragment?

```
int b [4] = {0 , 0 , 0 , 0};

MPI_Gather ( & rank , 1 , MPI_INT , b , 1 , MPI_INT , root
,MPI_COMM_WORLD);
```

Hint. The function prototype is as follows:

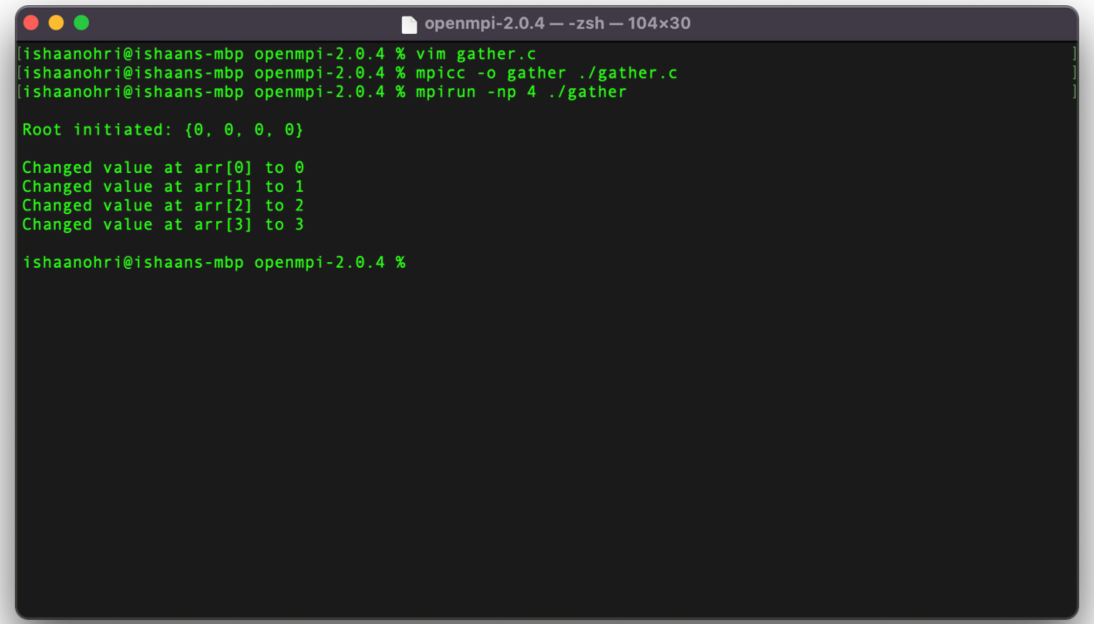
```
int MPI_Gather (
void * sendbuf ,       // pointer to send buffer
int sendcount ,             // number of items to send
MPI_Datatype sendtype ,    // type of send buffer data
void * recvbuf ,           // pointer to receive buffer
int recvcount ,            // items to receive per process
MPI_Datatype recvtype ,    // type of receive buffer data
int root ,            // rank of receiving process
MPI_Comm comm )       // MPI communicator to use
```

Source Code:



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

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

    int n, rank, world;
    int sum = 0;

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &world);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    int arr[] = {0, 0, 0, 0};

    if (rank == 3)
    {
        printf("\nRoot initiated: {0, 0, 0, 0}\n\n");
    }

    MPI_Gather(&rank, 1, MPI_INT, arr, 1, MPI_INT, 3, MPI_COMM_WORLD);

    if (rank == 3)
    {
        for (int i = 0; i < 4; i++)
        {
            printf("Changed value at arr[%d] to %d \n", i, arr[i]);
        }
        printf("\n");
    }

    MPI_Finalize();

    return 0;
}
~
```

Execution:



```
ishaanohri@ishaans-mbp openmpi-2.0.4 % vim gather.c
ishaanohri@ishaans-mbp openmpi-2.0.4 % mpicc -o gather ./gather.c
ishaanohri@ishaans-mbp openmpi-2.0.4 % mpirun -np 4 ./gather

Root initiated: {0, 0, 0, 0}

Changed value at arr[0] to 0
Changed value at arr[1] to 1
Changed value at arr[2] to 2
Changed value at arr[3] to 3

ishaanohri@ishaans-mbp openmpi-2.0.4 %
```

Remarks:

```
int MPI_Gather(const void *sendbuf, int sendcount,
MPI_Datatype sendtype, void *recvbuf, int recvcount,
MPI_Datatype recvtype, int root, MPI_Comm comm)
```

In the MPI_Gather command following are the meaning of all parameters:

sendbuf => starting address of send buffer (choice)
sendcount => number of elements in send buffer (integer)
sendtype => data type of send buffer elements (handle)
recvcount => number of elements for any single receive (integer, significant only at root)
recvtype => data type of recv buffer elements (significant only at root) (handle)
root => rank of receiving process (integer)
comm => communicator (handle)

Initially the array arr is declared as {0, 0, ,0 ,0} and then by the use of MPI_Gather we get the values changed to {0, 1, 2, 3}. Each and every root is sending its rank as the sendbuff. These received values are stored at the respective indices of the final root array. These value are displayed iteratively.