Roll No. 202CD005 **Sub: Introduction to Scalable Systems**
Name: Gavali Deshabhakt Nagnath

MPI Programming Assignment

1. MPI “Hello World” program :
Output:
Hello world from processor deshabhakt-pc, rank 0 out of 2 processors
Hello world from processor deshabhakt-pc, rank 1 out of 2 processors

2. Demonstration of MPI_Send() and MPI_Recv().
Output:
Value of x is : 0
Process 1 of 2, Value of x is 10
Source 0 Tag 1

3. Non-Blocking Send and Receive
Output:
Value of x is : 0
Process 1 of 2, Value of x is 0
Source 834106320 Tag 21912
Received i : 0
Received i : 1
Received i : 2
Received i : 3
Received i : 4

4. MPI_Send() standard mode:
Output:
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array x : 2
Received Array y : 1
Received Array y : 1
Received Array y : 1
Received Array y : 1
Received Array y : 1
Received Array y : 1

Received Array y : 1
Received Array y : 1
Received Array y : 1
Received Array y : 1

5. Demonstration of Broadcast operation : MPI_Bcast()
Output(With Single Processor):
2
Value of x in process 0 : 2

6. Demonstration of MPI_Reduce with Sum Operation
Output:
Value of y after reduce : 1

7. Demonstration of MPI_Gather()
Output:

Value of y[0] in process 0 : 10

Value of y[1] in process 0 : 10

8. Demonstration of MPI_Scatter()
Program:
/ 8. Demonstration of MPI_Scatter()

// • Note that the program is **ch**a**ng**ed to wo**rk** f**or** a**ny** num**b**er **of p**r**o**c**ess**es receiving two chunks from the array.

// • You may change according to what you want to explore.

```
#include <mpi.h>

#include <stdio.h>

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

{

   int size, myrank, x[8], y[2], i;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
   if (myrank == 0)
   {
      printf("Enter 8 values into array x:\n");
      for (i = 0; i < 8; i++)

         scanf("%d", &x[i]);
```

```
    }
    MPI_Scatter(&x, 8/size, MPI_INT, &y, 8/size, MPI_INT, 0, MPI_COMM_WORLD);
    for (i = 0; i < (8/size); i++)
        printf("\nValue of y in process %d : %d\n", myrank, y[i]);
    MPI_Finalize();
    return 0;
}
```

Output:

```
Enter 8 values into array x:
1
2
3
4
5
6
7
8

Value of y in process 0 : 1

Value of y in process 0 : 2

Value of y in process 1 : 3

Value of y in process 1 : 4

Value of y in process 2 : 5

Value of y in process 2 : 6

Value of y in process 3 : 7

Value of y in process 3 : 8
```

9. Write an MPI program to find the smallest element in a given array of size N.

Program:

```
#include <stdio.h>
#include <mpi.h>
int main(int argc, char **argv)
{
   int rank, size,n,lmin,gmin,x;
   int ArrIN[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24};
   int B[]={};
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   n=sizeof(ArrIN)/sizeof(int);
   x=n/size;
   MPI_Scatter(&ArrIN,x,MPI_INT,&B,x,MPI_INT,0,MPI_COMM_WORLD);
   lmin=B[0];
   for(int i=0;i<x;i++)
   {
      if(lmin>B[i])
      {
         lmin=B[i];
      }
   }
   printf("Local min of process %d is %d\n",rank,lmin);
   MPI_Reduce(&lmin,&gmin,size,MPI_INT,MPI_MIN,0,MPI_COMM_WORLD);
   if(rank==0)
   {
      printf("Min of Entire array is %d\n",gmin);
   }
   MPI_Finalize();
   return 0;
}
```

Output(with 4 processes):

Local min of process 2 is 13

Local min of process 3 is 19

Local min of process 0 is 1

Local min of process 1 is 7

Min of Entire array is 1

10. In a smart agriculture system in a large area like a state, sensors are deployed to collect temperature and humidity. The sensed information are stored in a server in the cloud. A query on calculating the average temperature and average humidity of the complete state needs the processing of 10 lakh data elements. Write a parallel program using MPI in which N number of processes run in parallel to calculate the average of 10 lakh elements stored in an array, in order to improve response time.

- Note: You may use number of elements to be smaller than 10 lakh for testing, as you have to initialize that many elements.

Program:

```
#include<stdio.h>
#include<mpi.h>
int main(int argv,char *argc[])
{
   int myrank,size,n,sum;
   MPI_Init(&argv,&argc);
   MPI_Comm_size(MPI_COMM_WORLD,&size);
   MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
   int A[]={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20};
   int B[]={};
   n=sizeof(A)/sizeof(int);
   int x=(n/size);
   MPI_Scatter(&A,x,MPI_INT,&B,x,MPI_INT,0,MPI_COMM_WORLD);
   int localsum=0;
```

```
    for(int i=0;i<x;i++)
    {
       localsum+=B[i];
    }
    MPI_Reduce(&localsum,&sum,1,MPI_FLOAT,MPI_SUM,0,MPI_COMM_WORLD);
    if(myrank==0)
    {
       float average = (float)sum/n;
       printf("Average of Array elements is %0.4f",average);
    }
    MPI_Finalize();
    return 0;
}
```

Output:

Average of Array elements is 10.5000