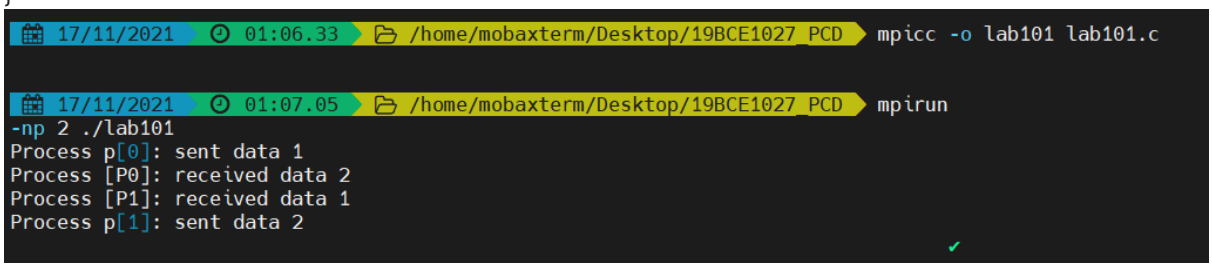
ARYAMAN MISHRA

19BCE1027

1. **Let Process 0 has variable A, and Process 1 has a variable B. Write MPI-like pseudocode to exchange these values between the processes. In other words, variable A should be shared to Process 1 and variable B should be shared to Process 0. Process 0 should display value of A and Process 1 should display value of B.**

```
#include <mpi.h>

#include <stdio.h>

int main(int argc, char **argv)

{

int rank;

int a, b, send_data;

const int root = 0;

MPI_Init(&argc, &argv);

MPI_Comm_rank (MPI_COMM_WORLD, &rank);

int world_size;

MPI_Comm_size(MPI_COMM_WORLD, &world_size);

if (rank != root)

{

b=2;

MPI_Recv(&a, 1, MPI_INT, rank-1, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE);

printf("Process [P%d]: received data %d\n", rank, a);

send_data=b;

}

else{

a=1;
```

```
send_data-a;

}

printf("Process p[%d]: sent data %d\n", rank, send_data);

MPI_Send(&send_data, 1, MPI_INT, (rank+1)%world_size,0, MPI_COMM_WORLD);

if (rank==root) {

MPI_Recv(&b, 1, MPI_INT, 1, 0, MPI_COMM_WORLD,MPI_STATUS_IGNORE);

printf("Process [P%d]: received data %d\n", rank, b);

}

MPI_Finalize();

return 0;

}
```



2. Create four processes P0, P1, P2 and P3. Let each process Pi sends its rank to another process Pj as given below. Let the receiving process Pj prints the sum of its rank and the rank received from Pi.

Sending Process Pi
Receiving Process Pj
P0
P1
P1
P2
P2
P3
P4
P0
*** You can assume any process to be P0, P1, P2 and P3**

```
#include <mpi.h>

#include <stdio.h>

int main(int argc, char **argv)

{
```

```
int rank,world_rank,rank1;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
if (world_rank != 0)
{
MPI_Recv(&rank1, 1, MPI_INT, world_rank - 1, 0,MPI_COMM_WORLD, MPI_STATUS_IGNORE);
rank=world_rank;
printf("Process P%d received rank %d from process P%d\n", world_rank,rank1, world_rank - 1);
printf("Sum of ranks=%d\n",(world_rank+rank1));
}
rank=world_rank;
MPI_Send(&rank, 1, MPI_INT, (world_rank + 1) % world_size,0, MPI_COMM_WORLD);
if (world_rank == 0)
rank=world_rank;
MPI_Recv(&rank1, 1, MPI_INT, world_size - 1, 0,MPI_COMM_WORLD, MPI_STATUS_IGNORE);
printf("Process P%d received rank %d from process P%d\n", world_rank,rank1, world_size - 1);
printf("Sum of ranks=%d\n",(world_rank+rank1));
MPI_Finalize();
}
```

```
/home/mobaxterm/Desktop/19BCE1027_PCD
17/11/2021  01:20.18  mpicc -o lab102 lab102.c


/home/mobaxterm/Desktop/19BCE1027_PCD
17/11/2021  01:22.40  mpirun -np 3 ./lab102
Process P0 received rank 2 from process P2
Sum of ranks=2
Process P1 received rank 0 from process P0
Sum of ranks=1
Process P2 received rank 1 from process P1
Sum of ranks=3
```

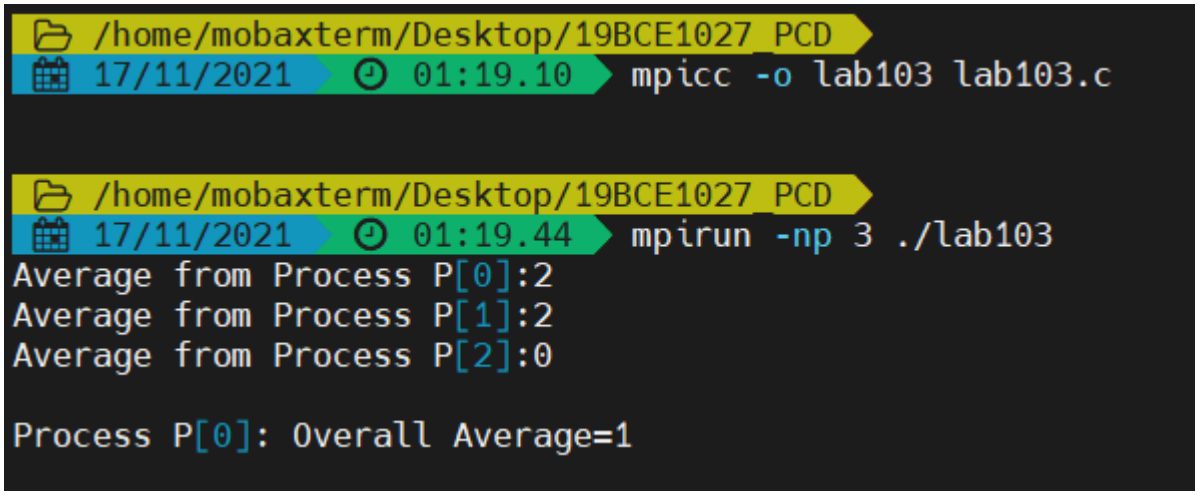
```
/home/mobaxterm/Desktop/19BCE1027_PCD
17/11/2021  01:23.32  mpirun -np 4 ./lab102
Process P0 received rank 3 from process P3
Sum of ranks=3
Process P1 received rank 0 from process P0
Sum of ranks=1
Process P2 received rank 1 from process P1
Sum of ranks=3
Process P3 received rank 2 from process P2
Sum of ranks=5
```

3. Consider four processes with their distributed integer data. Each process will have 'm' integer values where 'm' differs from one process to another. You can assume any value for 'm'. Let each individual process performs sum of their 'm' numbers and print it. Let each process will share its average (sum of 'm' numbers)/m to the process with rank 0. Let the process with rank 0 print the average of all the numbers received. In other words perform sum of the 4 data received and divide it by 4.

```
#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

int main(int argc, char **argv)

{

int rank, sum,overall_sum=0,i;

int avg;

int a[]={1,2,3},b[]={1,2,3,4},c[]={1,2,3,4,5},d[]={1,2,3,4,5,6};

MPI_Init(&argc, &argv);
```

```
MPI_Comm_rank(MPI_COMM_WORLD, &rank) ;
int world_size;
MPI_Comm_size(MPI_COMM_WORLD, &world_size);
int* sub_avgs=NULL;
if(rank==0)
{
sum=0;
sub_avgs = malloc(sizeof(int) * world_size);
for(i=0;i<3;i++)
{
sum=sum+a[i];
}
avg=sum/3;
MPI_Gather(&avg,1 , MPI_INT, sub_avgs, 1, MPI_INT, 0,MPI_COMM_WORLD) ;
for(i=0;i<world_size;i++)
{
printf("Average from Process P[%d]:%d\n",i, sub_avgs[i]);
overall_sum=overall_sum+sub_avgs[i];
}
printf("\nProcess P[%d]: Overall Average=%d\n", rank, (overall_sum/world_size));
}
else
{
sum=0;
if (rank==1)
{
for(i=0;i<4;i++)
{
sum=sum+b[i];
}
avg=sum/4;
```

```
MPI_Gather(&avg,1,MPI_INT,sub_avgs,1,MPI_INT,0,MPI_COMM_WORLD);
}
if (rank==2)
{
for(i=0;i<5;i++)
{
sum=sum+c[i];
}
avg-sum/5;
MPI_Gather(&avg,1 ,MPI_INT, sub_avgs, 1, MPI_INT, 0,MPI_COMM_WORLD) ;
}
if (rank==3)
{
for(i=0;i<6;i++)
{
sum=sum+c[i];
}
avg=sum/6;
MPI_Gather(&avg,1 , MPI_INT, sub_avgs, 1, MPI_INT, 0,MPI_COMM_WORLD) ;
}
}
MPI_Finalize();
}
```


/home/mobaxterm/Desktop/19BCE1027_PCD
17/11/2021 01:19.10 mpicc -o lab103 lab103.c
/home/mobaxterm/Desktop/19BCE1027_PCD
17/11/2021 01:19.44 mpirun -np 3 ./lab103
Average from Process P[0]:2
Average from Process P[1]:2
Average from Process P[2]:0
Process P[0]: Overall Average=1