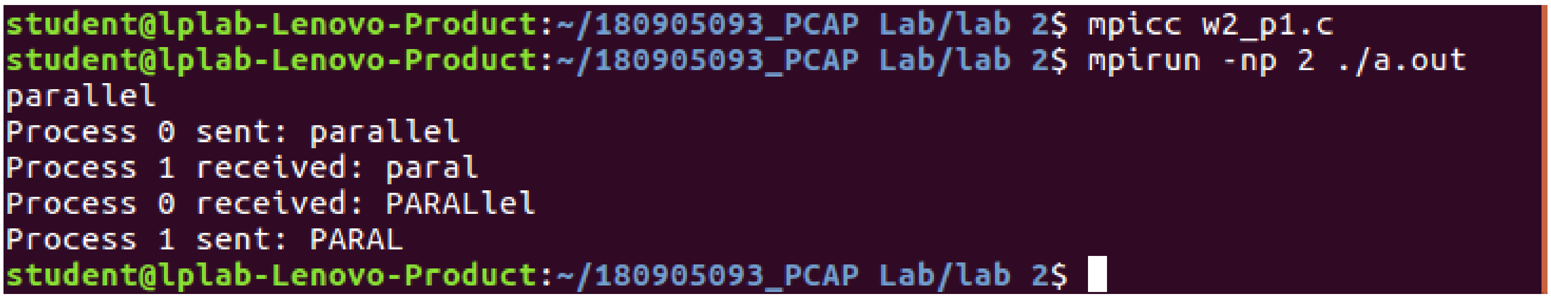
NAME : Rajvardhan Reddy Nandyala
Reg_No : 180905093
Section : B
Roll_No : 19 , Batch-1

PP_LAB – WEEK_2 :

P1)
```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int main (int argc, char *argv [])
{
int size, rank;
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
char word[5], y[5];
int len = 5*sizeof(char);
if (rank == 0)
{
scanf("%s", word);
MPI_Ssend(word, len, MPI_CHAR, 1, 101, MPI_COMM_WORLD);
printf("Process %d sent: %s\n", rank, word);
MPI_Recv(word, len, MPI_CHAR, 1, 102, MPI_COMM_WORLD,
&status);
printf("Process %d received: %s\n", rank, word);
}
else
{
MPI_Recv(y, len, MPI_CHAR, 0, 101, MPI_COMM_WORLD,
&status);
printf("Process %d received: %s\n", rank, y);for (int i = 0; i < strlen(y); i++)
{
if (y[i] >= 'A' && y[i] <= 'Z')
y[i] += 32;
else if (y[i] >= 'a' && y[i] <= 'z')
y[i] -= 32;
}
sleep(1);
MPI_Ssend(y, len, MPI_CHAR, 0, 102, MPI_COMM_WORLD);
printf("Process %d sent: %s\n", rank, y);
}
MPI_Finalize();
}
```

Output :

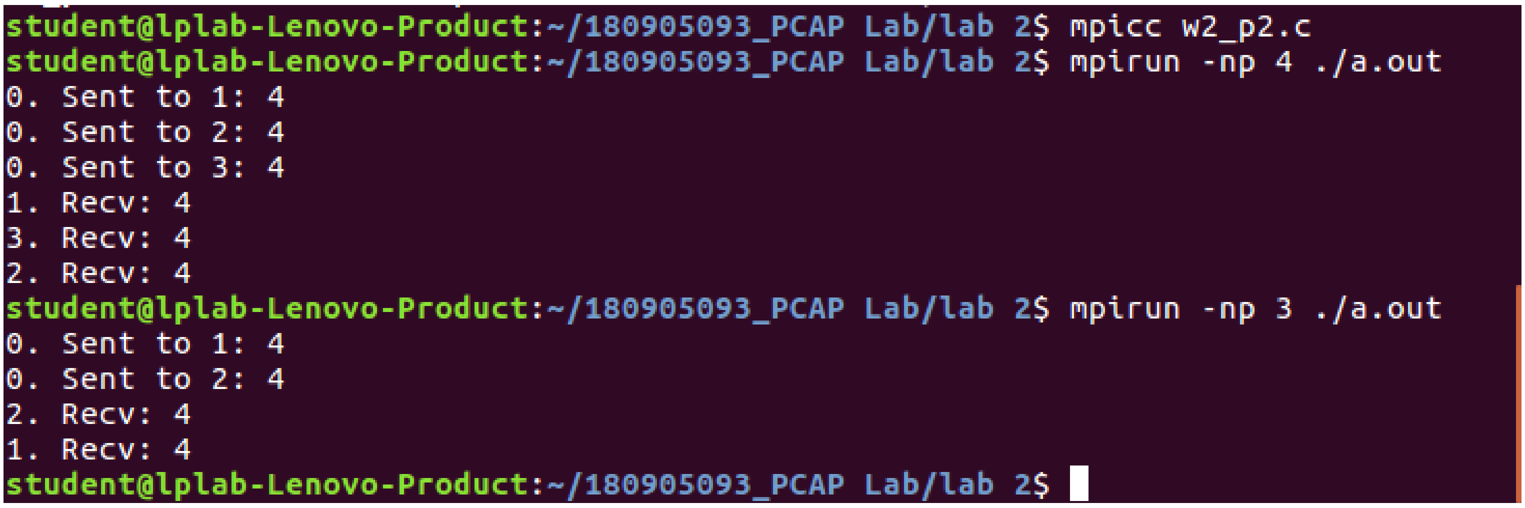


```
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpicc w2_p1.c
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpirun -np 2 ./a.out
parallel
Process 0 sent: parallel
Process 1 received: paral
Process 0 received: PARALlel
Process 1 sent: PARAL
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$
```

P2)

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

#define SIZE sizeof(int)

int main (int argc, char *argv [])
{
int size, rank;
MPI_Status status;

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

int *number = (int *)malloc(SIZE);

int i;

if (rank == 0)
{

*number = rand() % 10 + 1;

for (i = 1; i < size; ++i)
{
printf("%d. Sent to %d: %d\n", rank, i, *number);
// Send to the process with ID = i
MPI_Send(number, SIZE, MPI_INT, i, 100 + i,
MPI_COMM_WORLD);
}
}
else
{
// Revc from the process with ID = 0
MPI_Recv(number, SIZE, MPI_INT, 0, 100 + rank, MPI_COMM_WORLD,
&status);
printf("%d. Recv: %d\n", rank, *number);
}
MPI_Finalize();
}
```

Output :

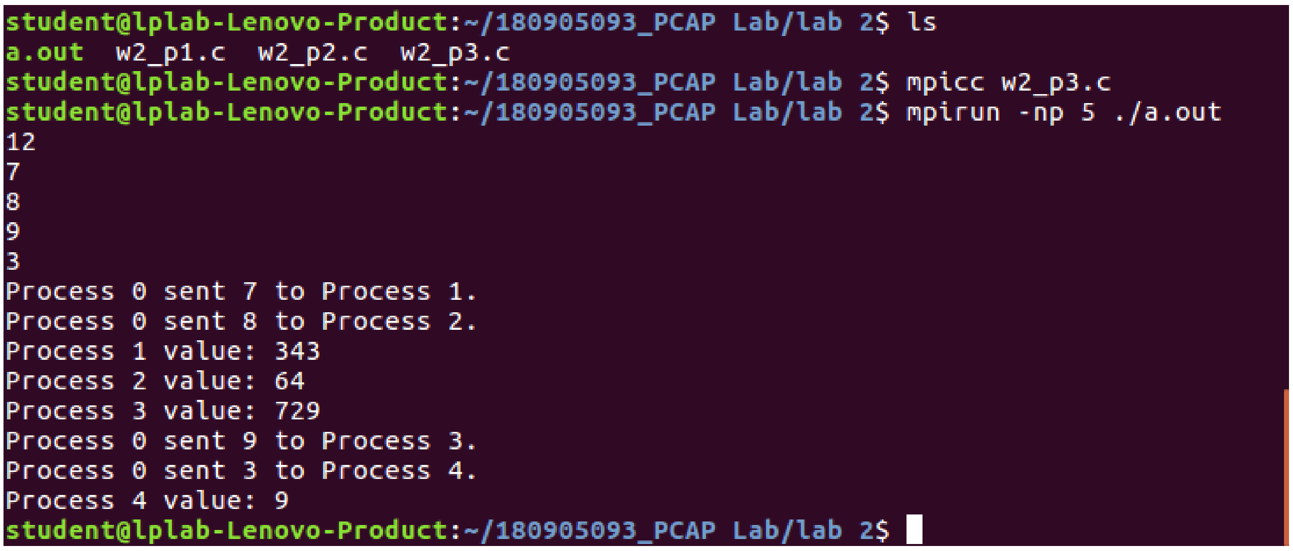


```
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpicc w2_p2.c
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpirun -np 4 ./a.out
0. Sent to 1: 4
0. Sent to 2: 4
0. Sent to 3: 4
1. Recv: 4
3. Recv: 4
2. Recv: 4
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpirun -np 3 ./a.out
0. Sent to 1: 4
0. Sent to 2: 4
2. Recv: 4
1. Recv: 4
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$
```

P3)

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[]){
int rank, size;
MPI_Status status;
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &size);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0)
{
int arr[5];
for(int i=0;i<5;i++)
scanf("%d", &arr[i]);
for (int i = 1; i < size; i++)
{
MPI_Ssend(arr + i, sizeof(int), MPI_INT, i, 100 + i, MPI_COMM_WORLD);
printf("Process %d sent %d to Process %d.\n", rank, arr[i], i);
}
}
else
{
int num;
MPI_Recv(&num, sizeof(int), MPI_INT, 0, 100 + rank, MPI_COMM_WORLD,
&status);
if (rank % 2 == 0)
num = num * num;
else
num = num * num * num;
printf("Process %d value: %d\n", rank, num);
}
MPI_Finalize();
}
```

Output :

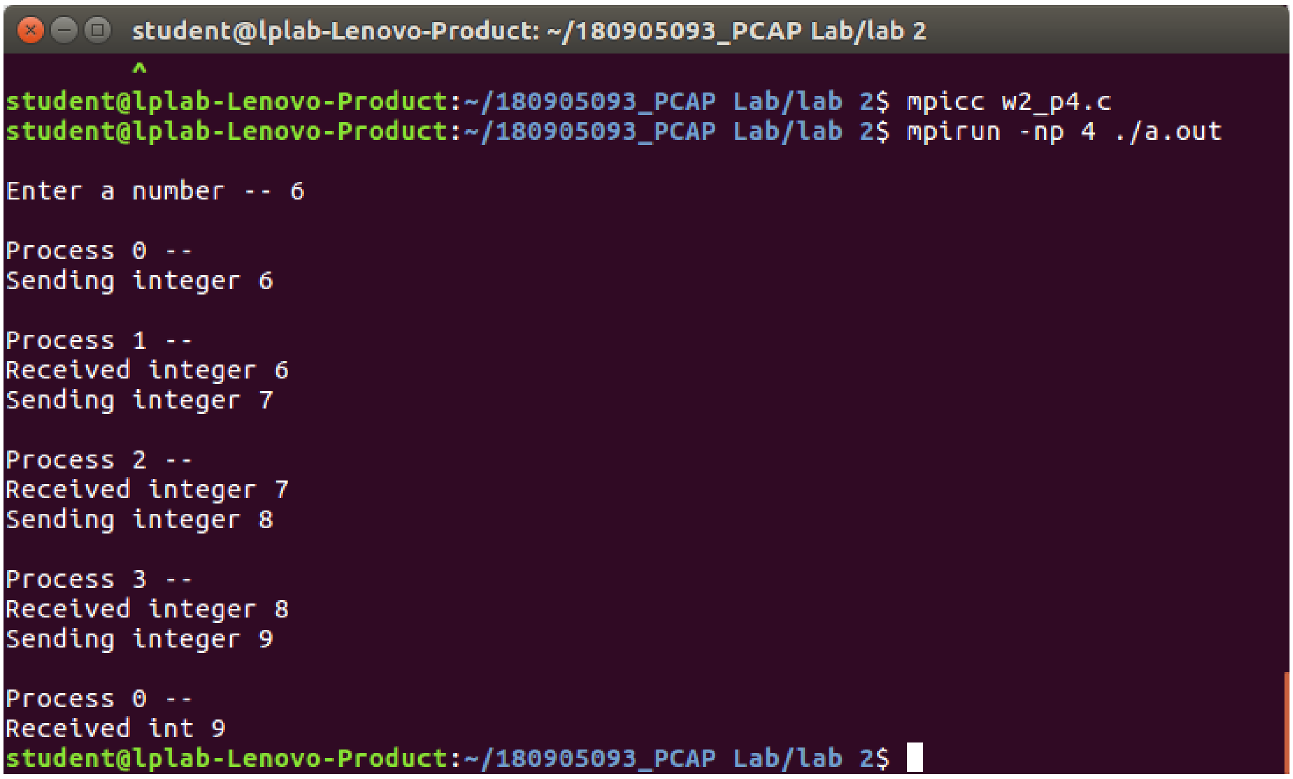


```
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ ls
a.out  w2_p1.c  w2_p2.c  w2_p3.c
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpicc w2_p3.c
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpirun -np 5 ./a.out
12
7
8
9
3
Process 0 sent 7 to Process 1.
Process 0 sent 8 to Process 2.
Process 1 value: 343
Process 2 value: 64
Process 3 value: 729
Process 0 sent 9 to Process 3.
Process 0 sent 3 to Process 4.
Process 4 value: 9
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$
```

P4)

```
#include<stdio.h>
#include<stdlib.h>
#include<mpi.h>
#include<string.h>

int main(int argc,char* argv[]){
    int rank,size,num;
    MPI_Init(&argc,&argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);
    if(rank==0){
        printf("\nEnter a number -- ");
        scanf("%d",&num);
        printf("\nProcess 0 -- ");
        printf("\nSending integer %d\n",num);
        MPI_Send(&num,1,MPI_INT,1,0,MPI_COMM_WORLD);
        MPI_Recv(&num,1,MPI_INT,size-
1,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
        printf("\nProcess 0 -- \nReceived int %d\n",num);
    }else{
        MPI_Recv(&num,1,MPI_INT,rank-
1,0,MPI_COMM_WORLD,MPI_STATUS_IGNORE);
        printf("\nProcess %d -- ",rank);
        printf("\nReceived integer %d",num);
        num++;
        printf("\nSending integer %d\n",num);
        MPI_Send(&num,1,MPI_INT,(rank+1)%size,0,MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}
```

Output :



```
student@lplab-Lenovo-Product: ~/180905093_PCAP Lab/lab 2
        ^
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpicc w2_p4.c
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$ mpirun -np 4 ./a.out

Enter a number -- 6

Process 0 --
Sending integer 6

Process 1 --
Received integer 6
Sending integer 7

Process 2 --
Received integer 7
Sending integer 8

Process 3 --
Received integer 8
Sending integer 9

Process 0 --
Received int 9
student@lplab-Lenovo-Product:~/180905093_PCAP Lab/lab 2$
```