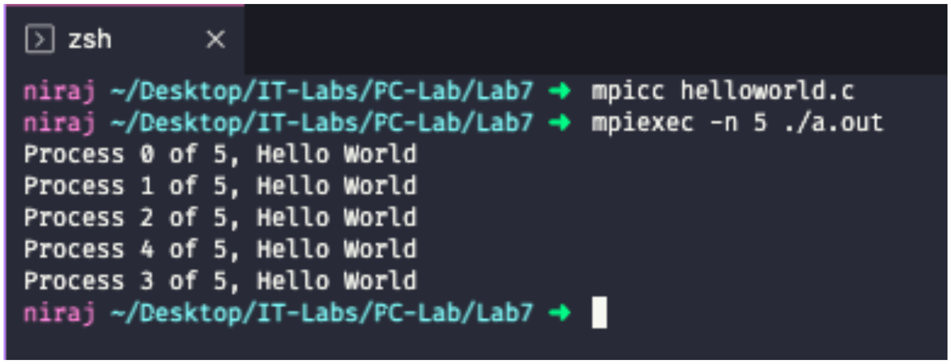
IT301 – Parallel Computing

Assignment – 7

Name: Niraj Nandish

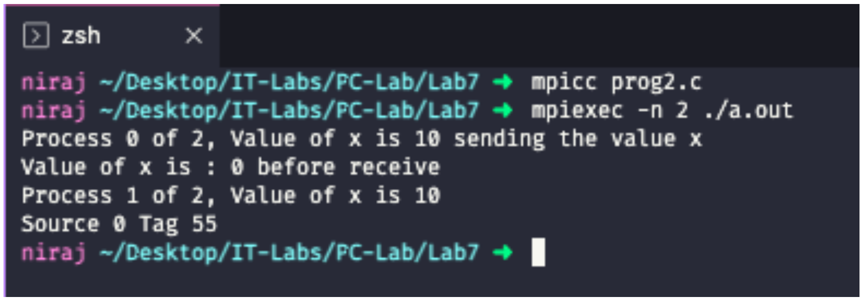
Roll No: 191IT234

1. Program 1 – Simple Hello World program



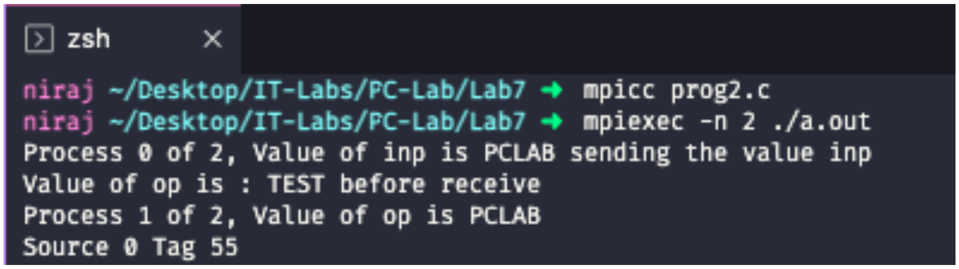
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc helloworld.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 5 ./a.out
Process 0 of 5, Hello World
Process 1 of 5, Hello World
Process 2 of 5, Hello World
Process 4 of 5, Hello World
Process 3 of 5, Hello World
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜
```

2. Program 2 – MPI_Send() and MPI_Recv()
    a. Source is process 0, destinations is process 1 and the tag is 55



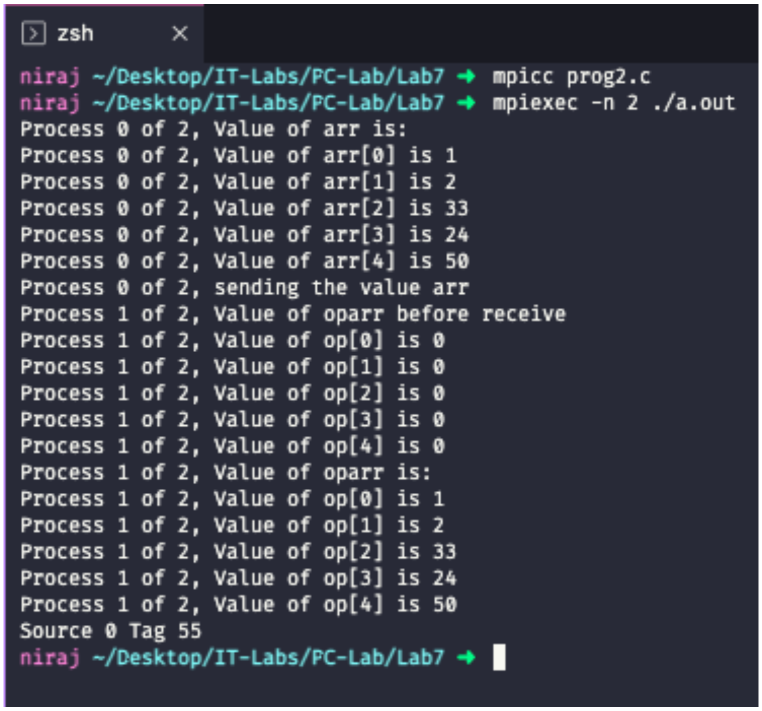
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog2.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
Process 0 of 2, Value of x is 10 sending the value x
Value of x is : 0 before receive
Process 1 of 2, Value of x is 10
Source 0 Tag 55
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜
```

b. Sending the string "PCLAB"



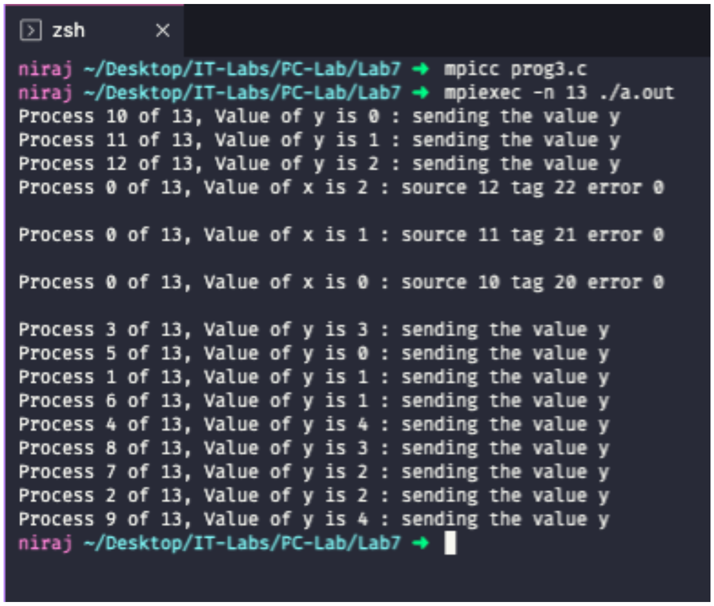
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog2.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
Process 0 of 2, Value of inp is PCLAB sending the value inp
Value of op is : TEST before receive
Process 1 of 2, Value of op is PCLAB
Source 0 Tag 55
```

c. Sending array of elements



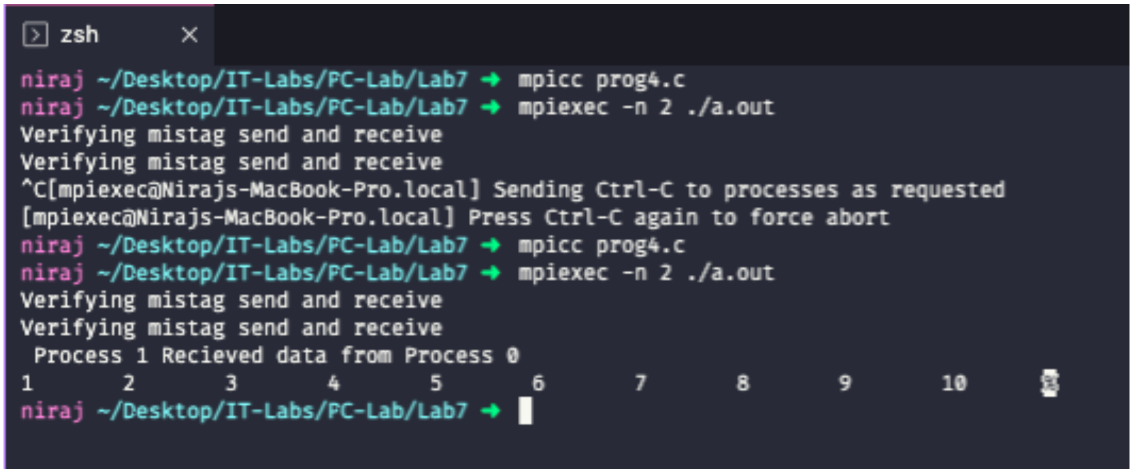
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog2.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
Process 0 of 2, Value of arr is:
Process 0 of 2, Value of arr[0] is 1
Process 0 of 2, Value of arr[1] is 2
Process 0 of 2, Value of arr[2] is 33
Process 0 of 2, Value of arr[3] is 24
Process 0 of 2, Value of arr[4] is 50
Process 0 of 2, sending the value arr
Process 1 of 2, Value of oparr before receive
Process 1 of 2, Value of op[0] is 0
Process 1 of 2, Value of op[1] is 0
Process 1 of 2, Value of op[2] is 0
Process 1 of 2, Value of op[3] is 0
Process 1 of 2, Value of op[4] is 0
Process 1 of 2, Value of oparr is:
Process 1 of 2, Value of op[0] is 1
Process 1 of 2, Value of op[1] is 2
Process 1 of 2, Value of op[2] is 33
Process 1 of 2, Value of op[3] is 24
Process 1 of 2, Value of op[4] is 50
Source 0 Tag 55
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ 
```

3. Program 3 – MPI_ANY_SOURCE and MPI_ANY_TAG
    a. Observation – In the MPI_Recv() function, the source parameter is given the value of MPI_ANY_SOURCE which means accept the data from any source and the tag parameter is given the value of MPI_ANY_TAG which means accept the data with any tag value.



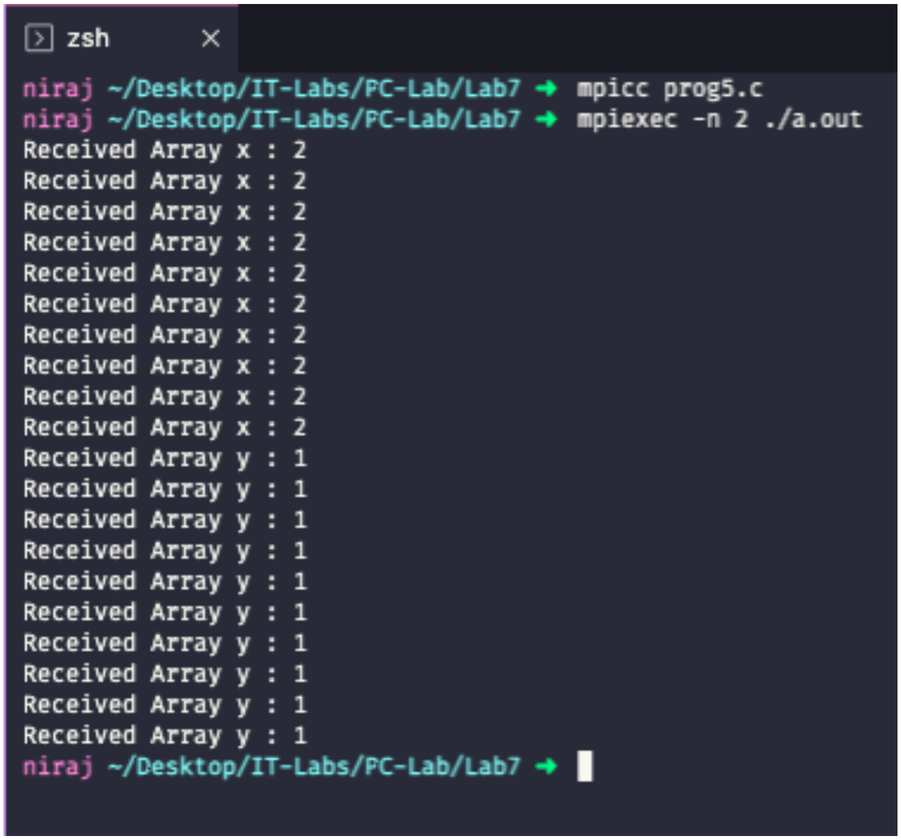
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog3.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 13 ./a.out
Process 10 of 13, Value of y is 0 : sending the value y
Process 11 of 13, Value of y is 1 : sending the value y
Process 12 of 13, Value of y is 2 : sending the value y
Process 0 of 13, Value of x is 2 : source 12 tag 22 error 0

Process 0 of 13, Value of x is 1 : source 11 tag 21 error 0

Process 0 of 13, Value of x is 0 : source 10 tag 20 error 0

Process 3 of 13, Value of y is 3 : sending the value y
Process 5 of 13, Value of y is 0 : sending the value y
Process 1 of 13, Value of y is 1 : sending the value y
Process 6 of 13, Value of y is 1 : sending the value y
Process 4 of 13, Value of y is 4 : sending the value y
Process 8 of 13, Value of y is 3 : sending the value y
Process 7 of 13, Value of y is 2 : sending the value y
Process 2 of 13, Value of y is 2 : sending the value y
Process 9 of 13, Value of y is 4 : sending the value y
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ █
```

4. Program 4 – MPI_Send() and MPI_Recv() with mismatched tag
    a. Observation – Since the tag value was different in MPI_Send() and MPI_Recv() functions, hence the data wasn't parsed in the receiving end. But when the tag value is corrected, it gets parsed, and we see the output.



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog4.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
Verifying mistag send and receive
Verifying mistag send and receive
^C[mpiexec@Nirajs-MacBook-Pro.local] Sending Ctrl-C to processes as requested
[mpiexec@Nirajs-MacBook-Pro.local] Press Ctrl-C again to force abort
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog4.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
Verifying mistag send and receive
Verifying mistag send and receive
 Process 1 Recieved data from Process 0
1       2       3       4       5       6       7       8       9       10      %
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ █
```

5. Program 5 - MPI_Send() and MPI_Recv() standard mode
    a. Observation – In process 0, we are first sending array "x" with the tag 1 and array "y" with the tag 2. In process 1, the first MPI_Recv() function waits for data with tag 2(Array "y" in P0) and stores that in array "x" and the second MPI_Recv() function waits for data with tag 1(Array "x" in P0) and stores that in array "y". So the first MPI_Send() function was blocked as the first MPI_Recv() function had a different tag value.



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpicc prog5.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜ mpiexec -n 2 ./a.out
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
niraj ~/Desktop/IT-Labs/PC-Lab/Lab7 ➜
```