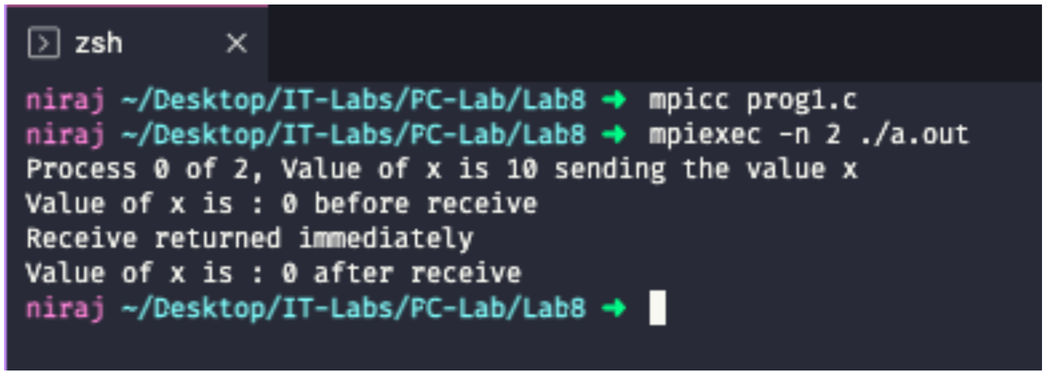
IT301 – Parallel Computing

Assignment – 8

Name: Niraj Nandish

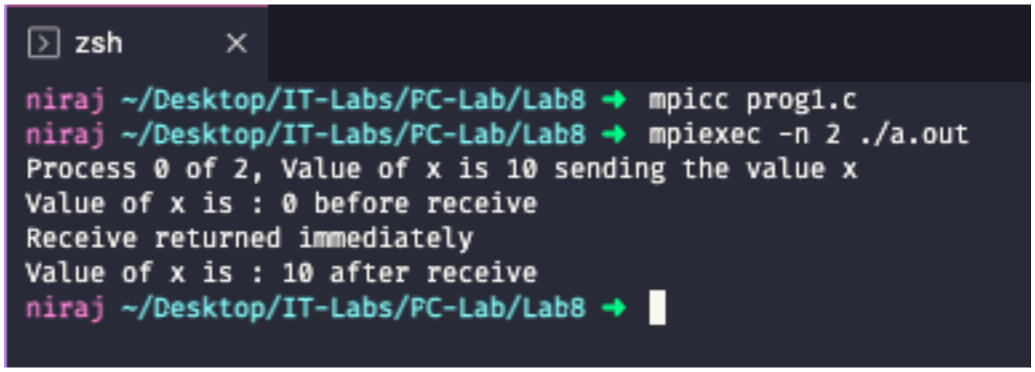
Roll No: 191IT234

1. Program 1 – MPI non-blocking Send and Receive
    a. Observation – Since we are using non-blocking send and receive here, the program executes without waiting to check if the data is sent/received completely.



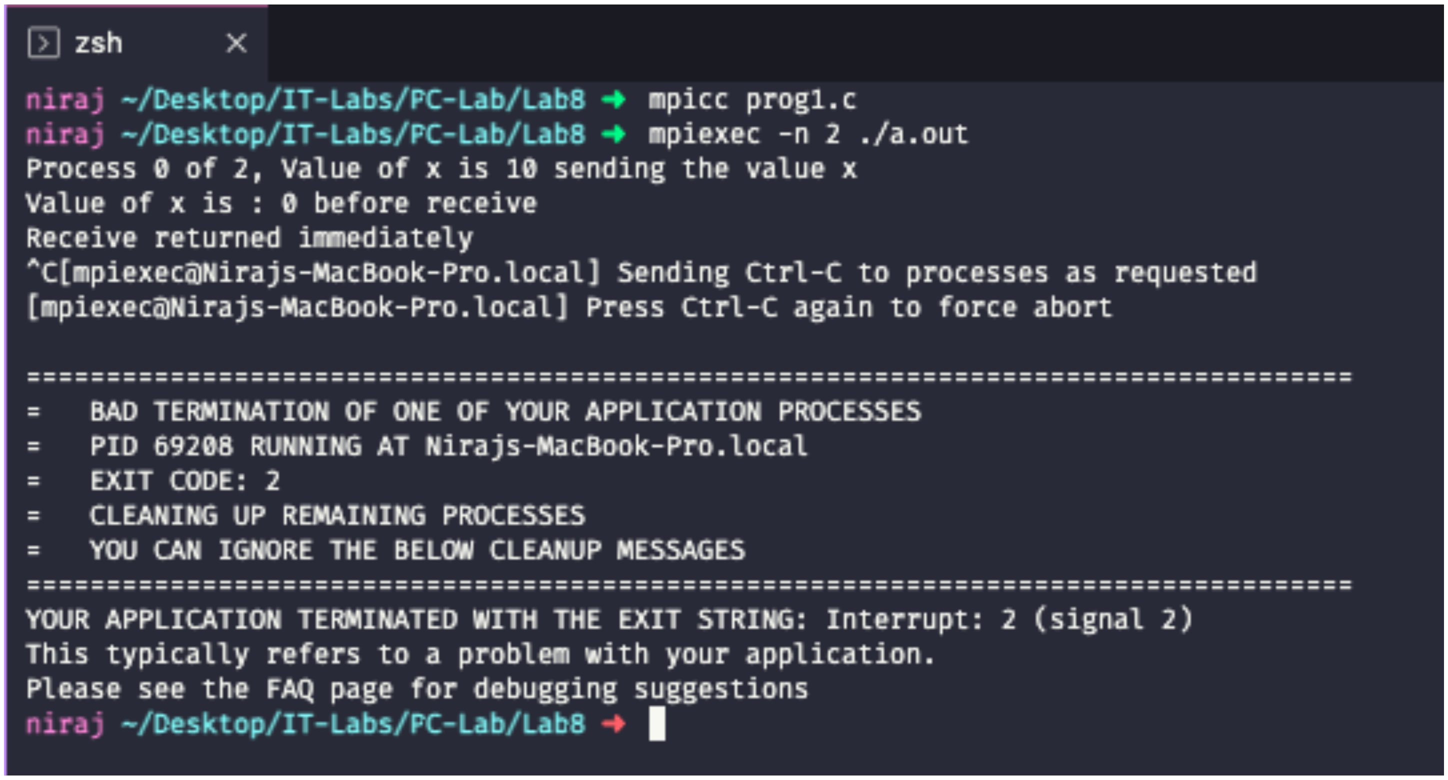
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog1.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpiexec -n 2 ./a.out
Process 0 of 2, Value of x is 10 sending the value x
Value of x is : 0 before receive
Receive returned immediately
Value of x is : 0 after receive
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

    b. Observation – Here the program waits for the data to be sent/received completely because of the MPI_wait() function



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog1.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ 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
Receive returned immediately
Value of x is : 10 after receive
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

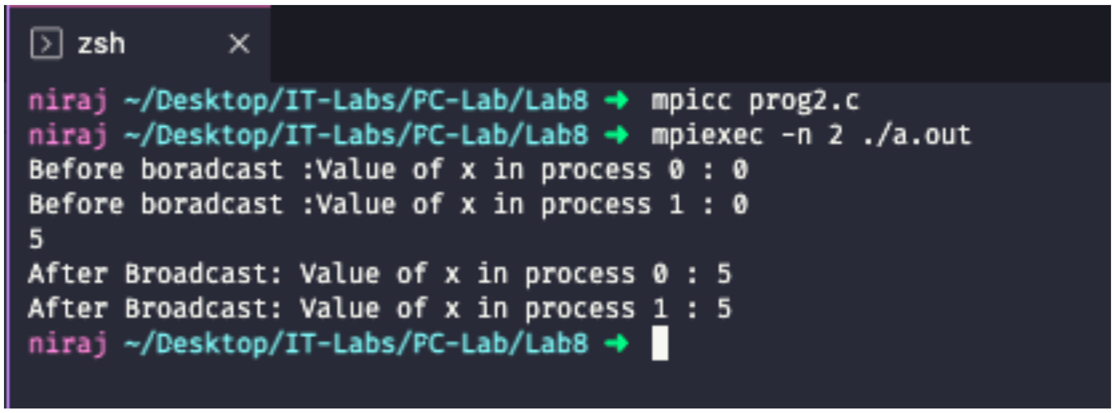
c. Observation – When the tag number is different in the sending and receiving process, the MPI_Wait() function makes the receiving process wait as it is expecting data from the sending process. Hence, this leads to a deadlock in the program.



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜  mpicc prog1.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜  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
Receive returned immediately
^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

===================================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   PID 69208 RUNNING AT Nirajs-MacBook-Pro.local
=   EXIT CODE: 2
=   CLEANING UP REMAINING PROCESSES
=   YOU CAN IGNORE THE BELOW CLEANUP MESSAGES
===================================================================================
YOUR APPLICATION TERMINATED WITH THE EXIT STRING: Interrupt: 2 (signal 2)
This typically refers to a problem with your application.
Please see the FAQ page for debugging suggestions
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

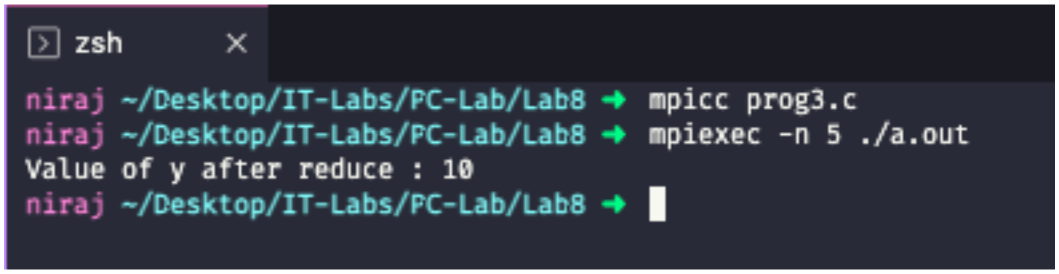
2. Program 2 – Bcast()

    a. Observation – The Bcast() function broadcasts the given data in the root process to all other processes.



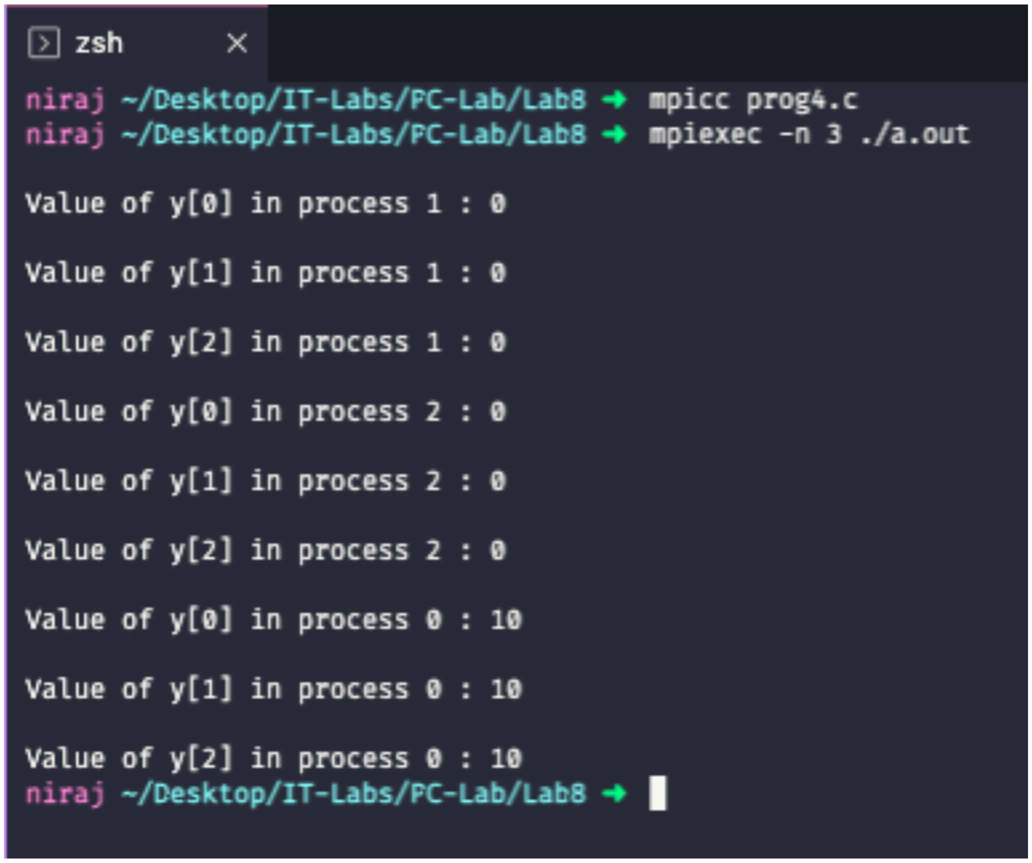
```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜  mpicc prog2.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜  mpiexec -n 2 ./a.out
Before boradcast :Value of x in process 0 : 0
Before boradcast :Value of x in process 1 : 0
5
After Broadcast: Value of x in process 0 : 5
After Broadcast: Value of x in process 1 : 5
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

3. Program 3 – Reduce()
   a. Observation – The Reduce() function reduces the given data from all processes according to the given operator and stores the reduced data in the root process. Here the operation was "sum" and the data in each process was its rank, so the final value after reduction for 5 processes is 0+1+2+3+4 = 10.



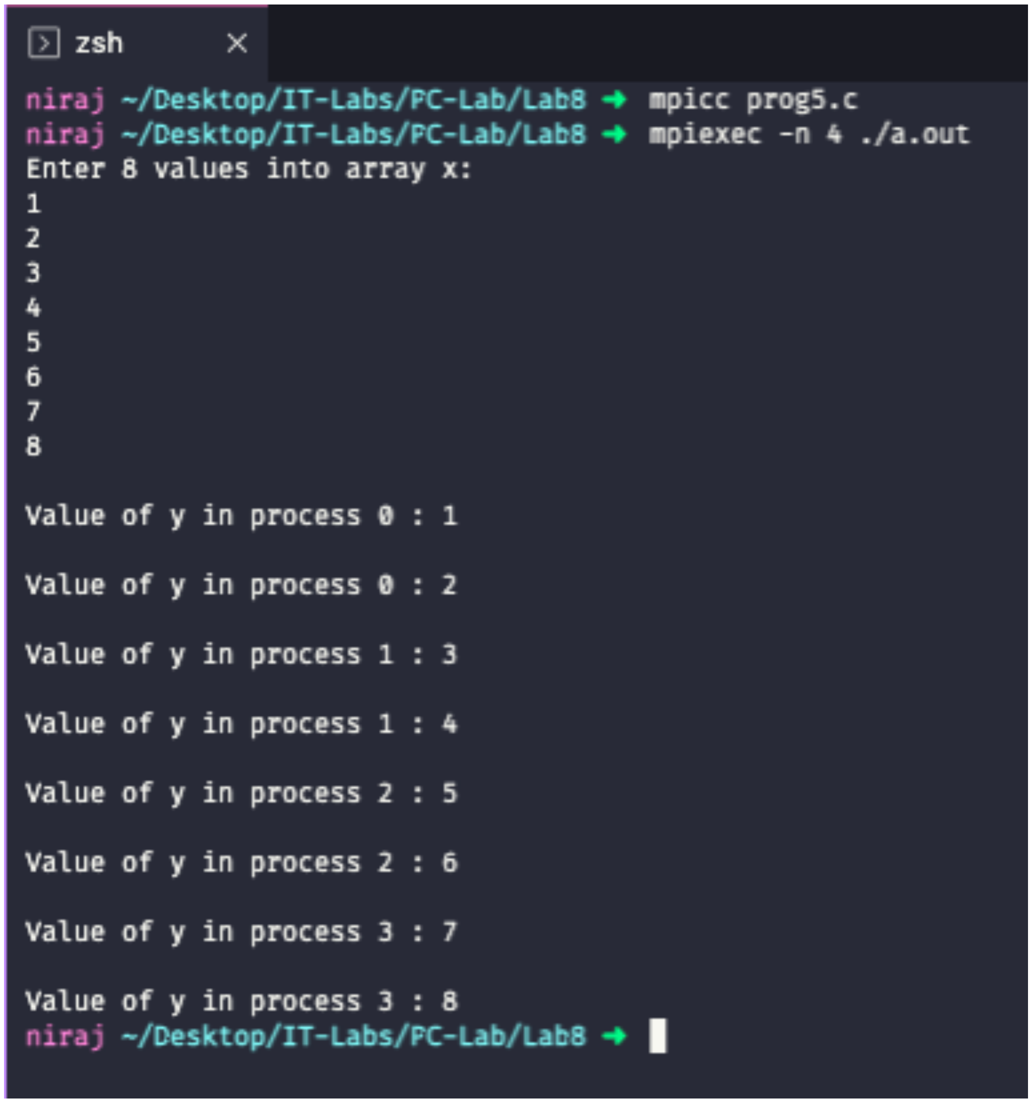
```
zsh
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog3.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpiexec -n 5 ./a.out
Value of y after reduce : 10
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

4. Program 4 – MPI_Gather()
   a. Observation – The MPI_Gather() function collects the given data from all processes and stores them in a given buffer at the specified process.



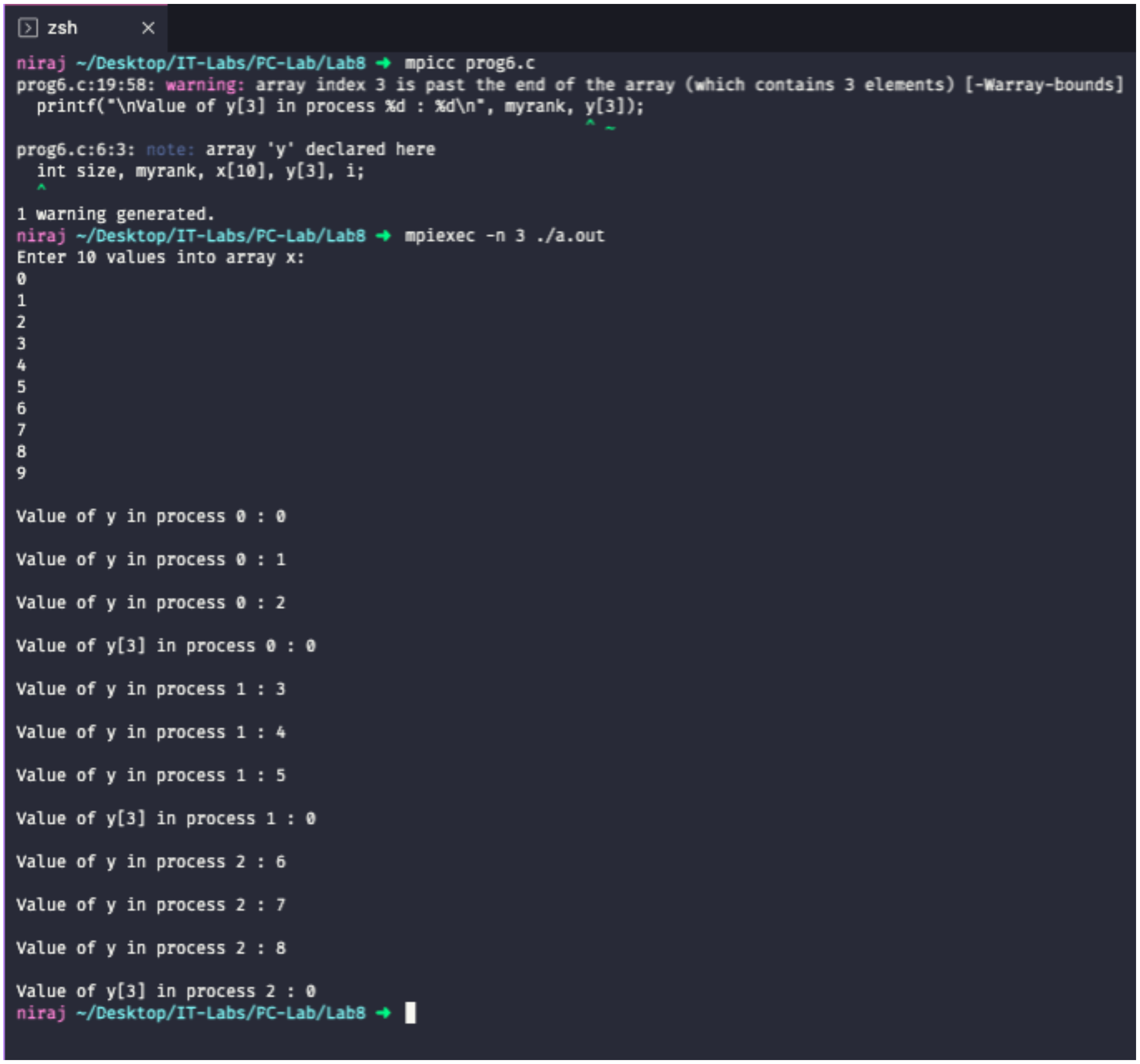
```
zsh
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog4.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpiexec -n 3 ./a.out

Value of y[0] in process 1 : 0

Value of y[1] in process 1 : 0

Value of y[2] in process 1 : 0

Value of y[0] in process 2 : 0

Value of y[1] in process 2 : 0

Value of y[2] in process 2 : 0

Value of y[0] in process 0 : 10

Value of y[1] in process 0 : 10

Value of y[2] in process 0 : 10
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

5. Program 5 – MPI_Scatter()
    a. Observation – The MPI_Scatter() function sends chunks in a buffer from the specified process to all processes. Here we input numbers from 1 to 8 and the values are equally distributed among the 4 processes.



```
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog5.c
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpiexec -n 4 ./a.out
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
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```

6. Program 6 – MPI_Scatter() with partial scatter
    a. Observation – As the process that sends data has more data so the program distributes the present data among the processes and remaining data is discarded.



```
zsh
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpicc prog6.c
prog6.c:19:58: warning: array index 3 is past the end of the array (which contains 3 elements) [-Warray-bounds]
  printf("\nValue of y[3] in process %d : %d\n", myrank, y[3]);
                                                         ^ ~

prog6.c:6:3: note: array 'y' declared here
  int size, myrank, x[10], y[3], i;
  ^

1 warning generated.
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜ mpiexec -n 3 ./a.out
Enter 10 values into array x:
0
1
2
3
4
5
6
7
8
9

Value of y in process 0 : 0

Value of y in process 0 : 1

Value of y in process 0 : 2

Value of y[3] in process 0 : 0

Value of y in process 1 : 3

Value of y in process 1 : 4

Value of y in process 1 : 5

Value of y[3] in process 1 : 0

Value of y in process 2 : 6

Value of y in process 2 : 7

Value of y in process 2 : 8

Value of y[3] in process 2 : 0
niraj ~/Desktop/IT-Labs/PC-Lab/Lab8 ➜
```