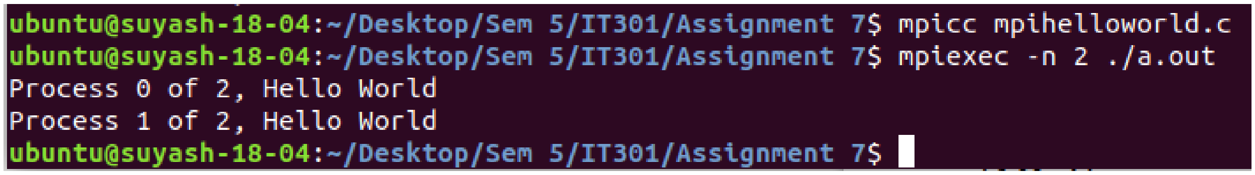
IT301 Assignment 7

NAME: SUYASH CHINTAWAR
ROLL NO.: 191IT109
TOPIC: MPI PROGRAMMING

NOTE: Code for problems 1,2(a),3,4,5 have not been attached as already provided.

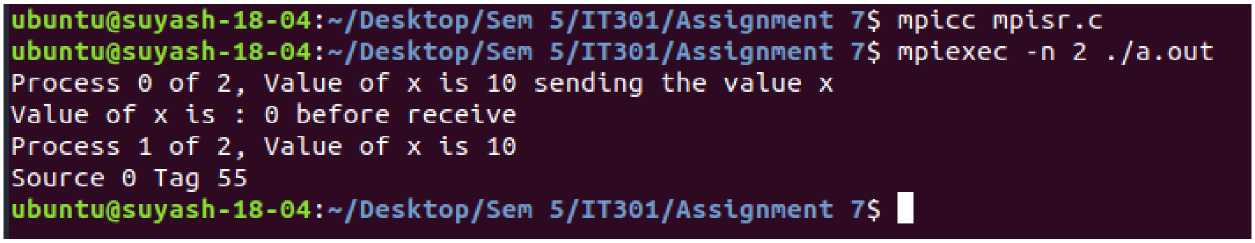
Q1. Simple Hello World program to find the rank and size of the communication world.
SOLUTION:
Output:



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc mpihelloworld.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 2 ./a.out
Process 0 of 2, Hello World
Process 1 of 2, Hello World
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Size of communication world: 2
Rank of processes : 0 and 1.

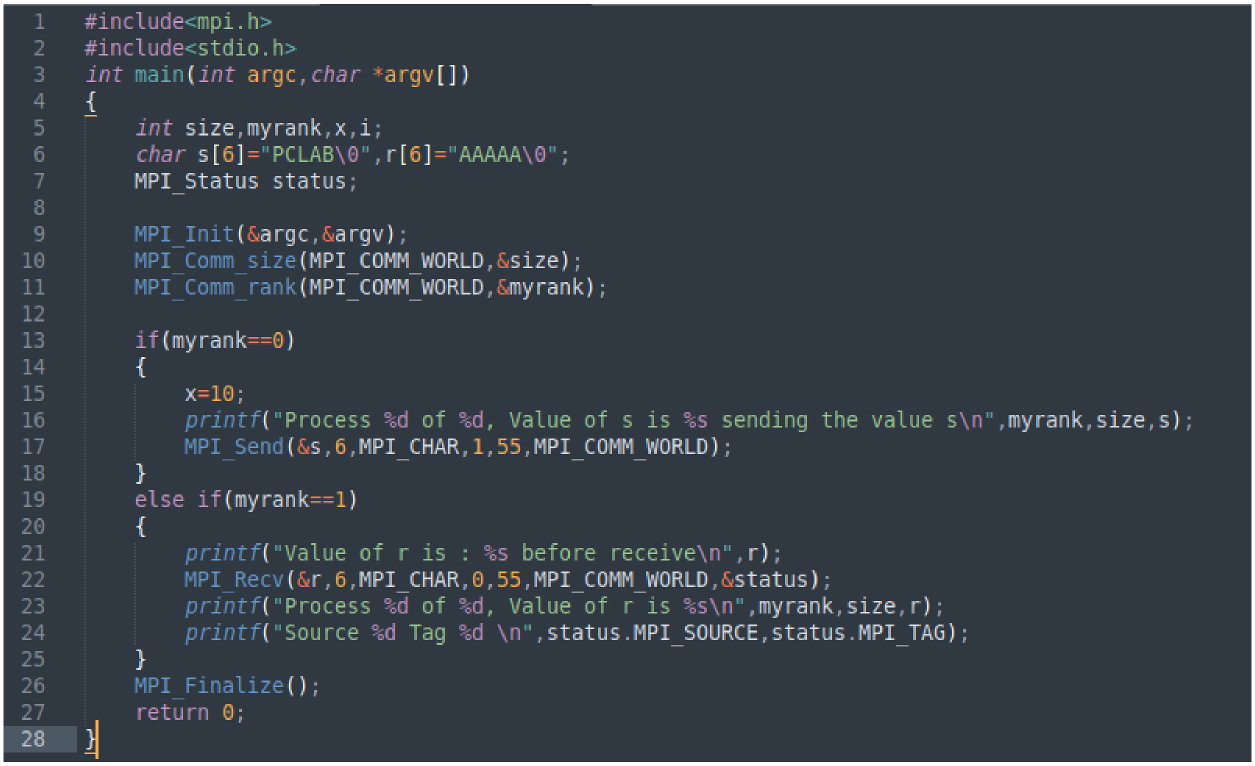
Q2. MPI_Send() and MPI_Recv() for sending an integer.
(a) Note down source , destination and tag.
(b) Modify the program to send the string “PCLAB” and add a screenshot of the result.
c) Modify the program to send an array of elements and add a screenshot of the result.
SOLUTION:
(a) Output:



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc mpisr.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 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
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

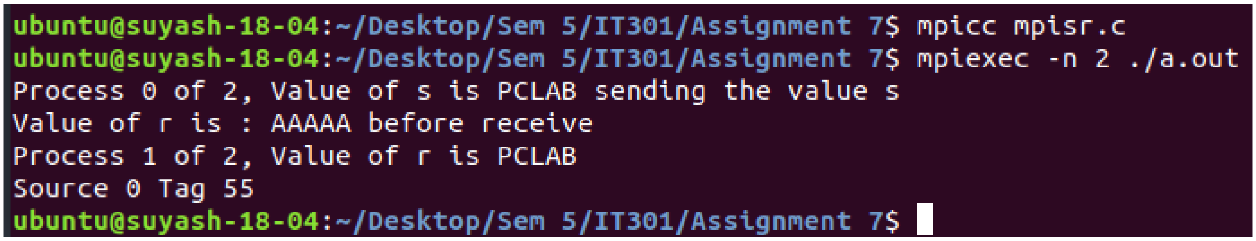
Source: Process 0
Destination: Process 1
Tag: 55

(b) Program:



```c
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[])
{
    int size,myrank,x,i;
    char s[6]="PCLAB\0",r[6]="AAAAA\0";
    MPI_Status status;

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

    if(myrank==0)
    {
        x=10;
        printf("Process %d of %d, Value of s is %s sending the value s\n",myrank,size,s);
        MPI_Send(&s,6,MPI_CHAR,1,55,MPI_COMM_WORLD);
    }
    else if(myrank==1)
    {
        printf("Value of r is : %s before receive\n",r);
        MPI_Recv(&r,6,MPI_CHAR,0,55,MPI_COMM_WORLD,&status);
        printf("Process %d of %d, Value of r is %s\n",myrank,size,r);
        printf("Source %d Tag %d \n",status.MPI_SOURCE,status.MPI_TAG);
    }
    MPI_Finalize();
    return 0;
}
```

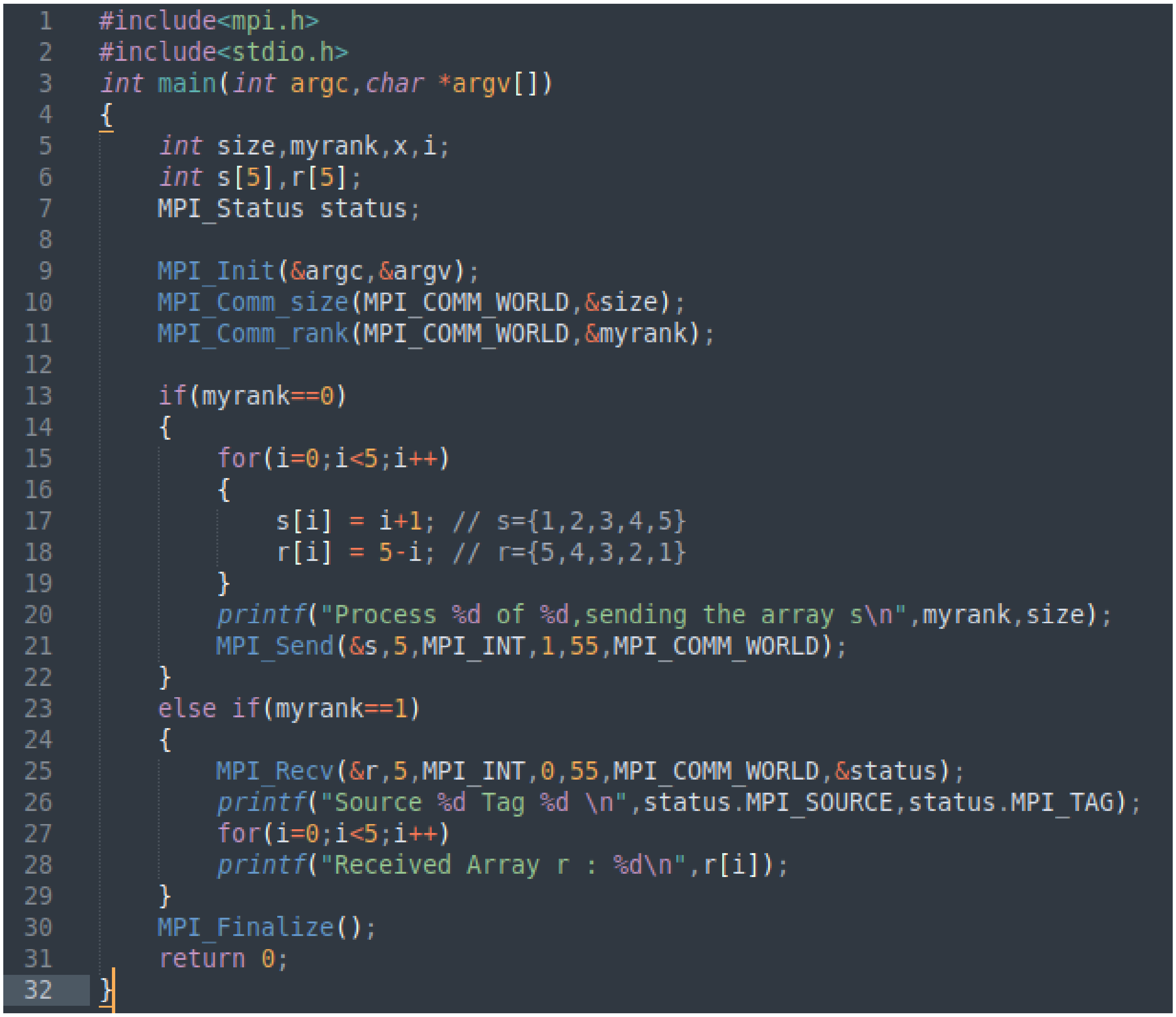
Output:



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc mpisr.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 2 ./a.out
Process 0 of 2, Value of s is PCLAB sending the value s
Value of r is : AAAAA before receive
Process 1 of 2, Value of r is PCLAB
Source 0 Tag 55
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$
```

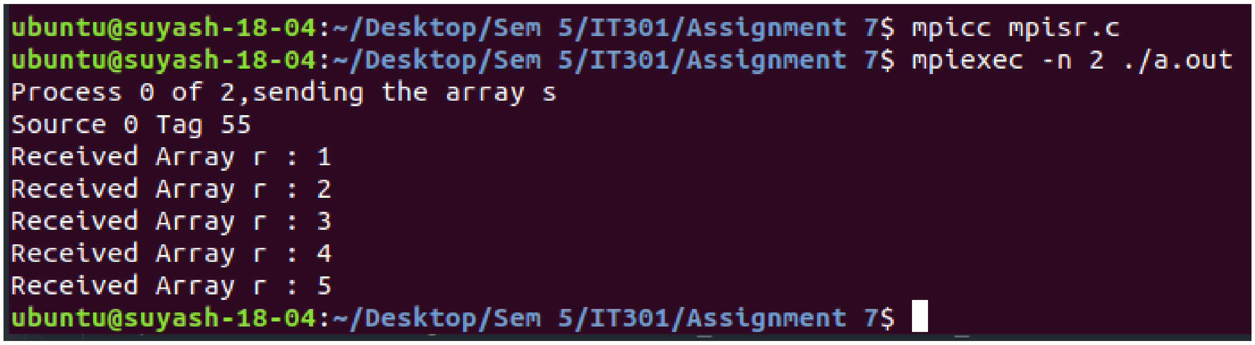
(c) Program:

(continued...)



```c
#include<mpi.h>
#include<stdio.h>
int main(int argc,char *argv[])
{
    int size,myrank,x,i;
    int s[5],r[5];
    MPI_Status status;

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

    if(myrank==0)
    {
        for(i=0;i<5;i++)
        {
            s[i] = i+1; // s={1,2,3,4,5}
            r[i] = 5-i; // r={5,4,3,2,1}
        }
        printf("Process %d of %d,sending the array s\n",myrank,size);
        MPI_Send(&s,5,MPI_INT,1,55,MPI_COMM_WORLD);
    }
    else if(myrank==1)
    {
        MPI_Recv(&r,5,MPI_INT,0,55,MPI_COMM_WORLD,&status);
        printf("Source %d Tag %d \n",status.MPI_SOURCE,status.MPI_TAG);
        for(i=0;i<5;i++)
        printf("Received Array r : %d\n",r[i]);
    }
    MPI_Finalize();
    return 0;
}
```

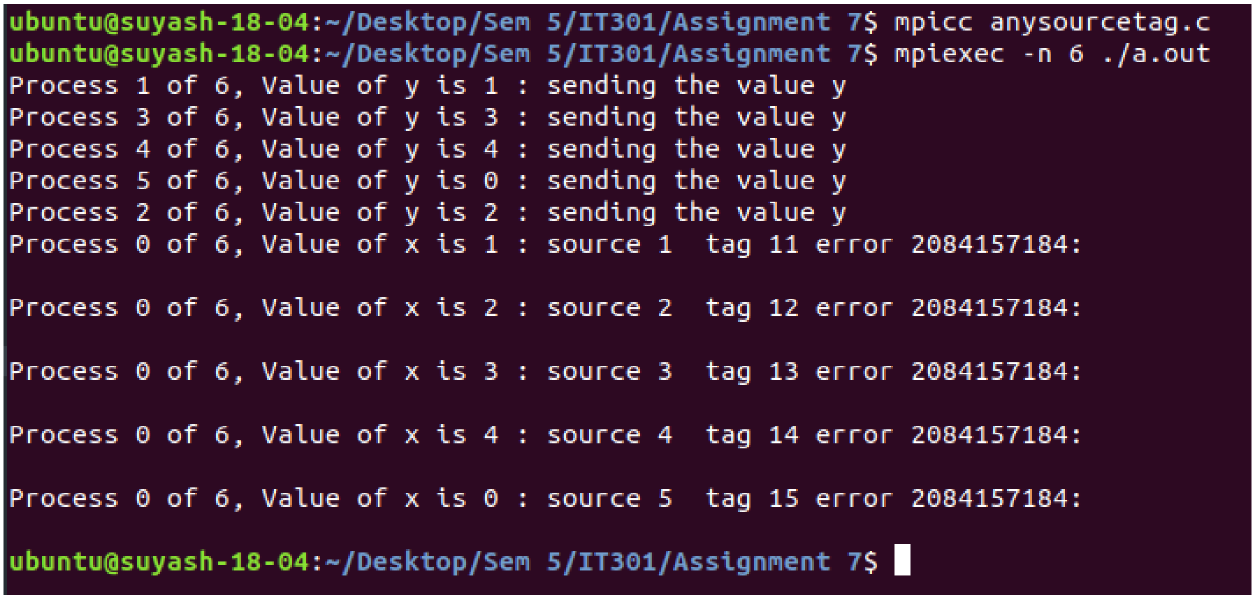
Output:



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc mpisr.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 2 ./a.out
Process 0 of 2,sending the array s
Source 0 Tag 55
Received Array r : 1
Received Array r : 2
Received Array r : 3
Received Array r : 4
Received Array r : 5
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Q3. MPI_Send() and MPI_Recv() with MPI_ANY_SOURCE, MPI_ANY_TAG. Note down the results and write your observation.

Output:

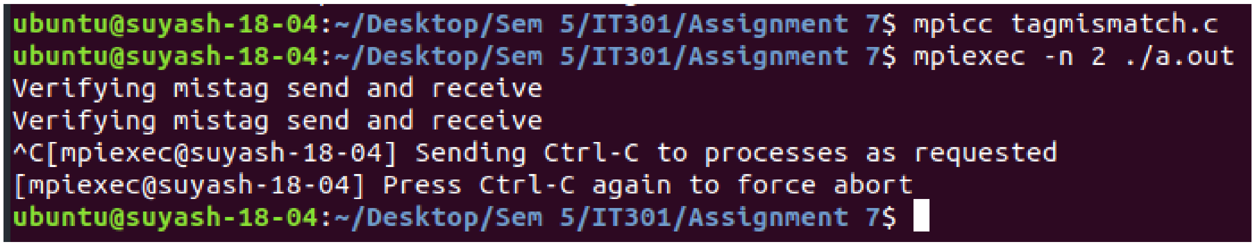


```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc anysourcetag.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 6 ./a.out
Process 1 of 6, Value of y is 1 : sending the value y
Process 3 of 6, Value of y is 3 : sending the value y
Process 4 of 6, Value of y is 4 : sending the value y
Process 5 of 6, Value of y is 0 : sending the value y
Process 2 of 6, Value of y is 2 : sending the value y
Process 0 of 6, Value of x is 1 : source 1  tag 11 error 2084157184:

Process 0 of 6, Value of x is 2 : source 2  tag 12 error 2084157184:

Process 0 of 6, Value of x is 3 : source 3  tag 13 error 2084157184:

Process 0 of 6, Value of x is 4 : source 4  tag 14 error 2084157184:

Process 0 of 6, Value of x is 0 : source 5  tag 15 error 2084157184:

ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Observation: We can see that the print statements are not necessarily in order. Process 0 is always receiving and other processes are sending data. The error is a garbage value as there is no error. The sending buffer is the variable ‘y’ and the receiving buffer is the variable ‘x’.

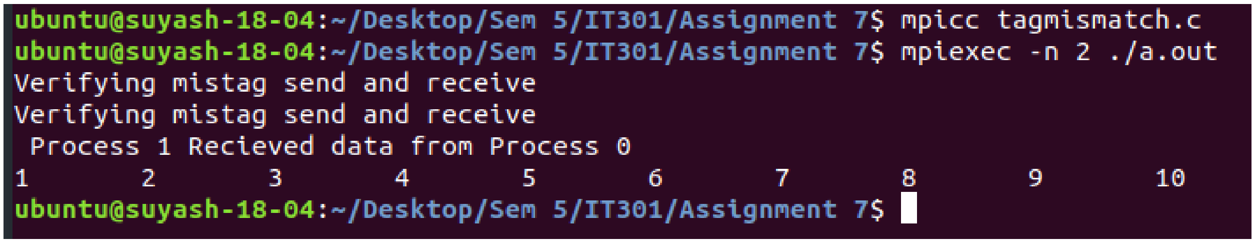
Q4. MPI_Send() and MPI_Recv() with mismatched tag. Record the result for mismatched tag and also after correcting tag value of send receive as same number

Output (With mismatched tag):



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc tagmismatch.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 2 ./a.out
Verifying mistag send and receive
Verifying mistag send and receive
^C[mpiexec@suyash-18-04] Sending Ctrl-C to processes as requested
[mpiexec@suyash-18-04] Press Ctrl-C again to force abort
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Output (After correcting the program):

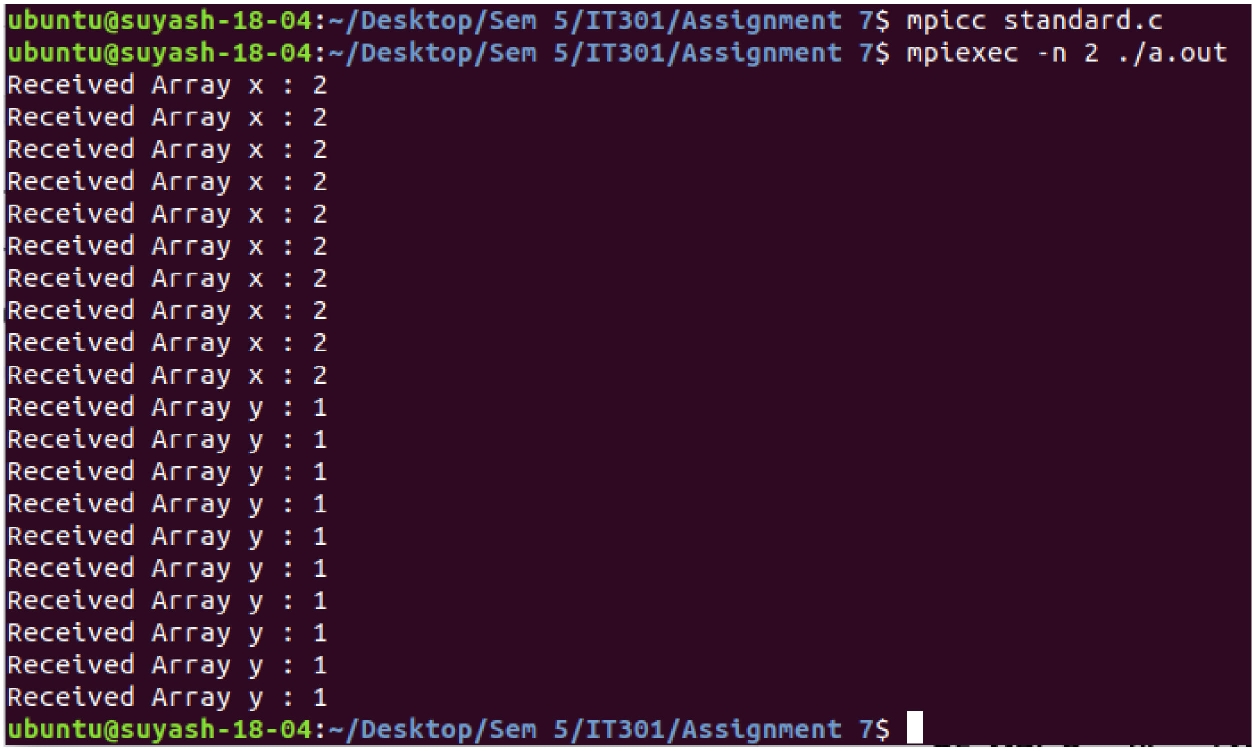


```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc tagmismatch.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpiexec -n 2 ./a.out
Verifying mistag send and receive
Verifying mistag send and receive
 Process 1 Recieved data from Process 0
1       2       3       4       5       6       7       8       9       10
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Observation: The program doesn’t stop when the tag is mismatched.

Q5. MPI_Send() and MPI_Recv() standard mode:
Note down your observation on the content of x and y at Process 1 and Explain the importance of tag.

Output:



```
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ mpicc standard.c
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 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
ubuntu@suyash-18-04:~/Desktop/Sem 5/IT301/Assignment 7$ 
```

Observation:

We can see that the arrays ‘x’ and ‘y’ have got swapped. This happens because when the first array x is sent, it doesn’t find a matching receiver so the send will return after copying the data in the system buffer. Then array y is sent as usual and it finds a matching receiver as well. But here, the receiving buffer is x so the values of array y are copied into x and the same happens for the other transaction as well.

THANK YOU