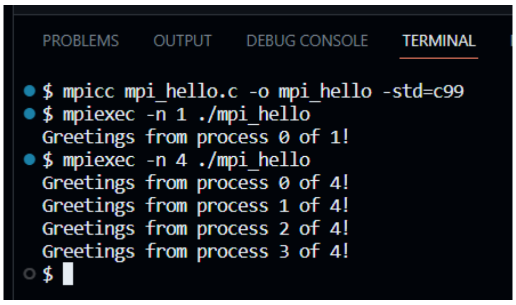
Assignment 1 Milestone Report

Steps

1. Compile and run the program to replicate the output given in the textbook. First with one process, then with four.

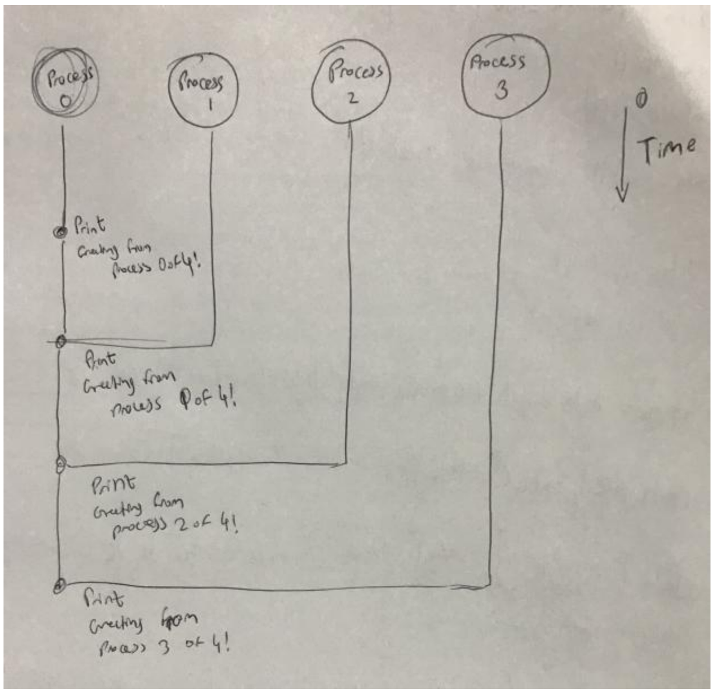


```
● $ mpicc mpi_hello.c -o mpi_hello -std=c99
● $ mpiexec -n 1 ./mpi_hello
  Greetings from process 0 of 1!
● $ mpiexec -n 4 ./mpi_hello
  Greetings from process 0 of 4!
  Greetings from process 1 of 4!
  Greetings from process 2 of 4!
  Greetings from process 3 of 4!
○ $
```

2. Write a brief and precise explanation of how the program works.

Program has two main functions of MPI_Send and MPI_Recv. MPI_Send will send the string from one process to another. While, MPI_Recv will receive it and displays it. The program prints 'Greetings' from each of the process. The 1! and 4! represents how many times to run/number of processes.

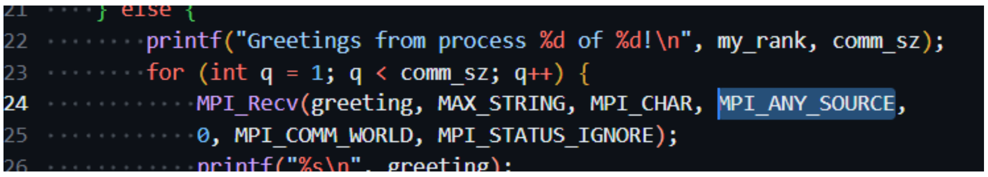
3. Draw a diagram showing all of the messages sent when executed with four tasks. Number the messages in the order in which they occur.


Process 0
Process 1
Process 2
Process 3
0
Time
Print Greeting from Process 0 of 4!
Print Greeting from Process 1 of 4!
Print Greeting from process 2 of 4!
Print Greeting from Process 3 of 4!

4. Question: Will the messages be ordered in the same way for all executions of the program? Explain.

The messages will be ordered the same way for all the executions. This is because of the for-loop in MPI_Send which have an increment beginning from 0. The for loop in MPI_Recv just makes it print whatever order is presented from the MPI_Send.

5. Create a second version of the program with only one change. This change is to use the special constant MPI_ANY_SOURCE in the call to Receive.

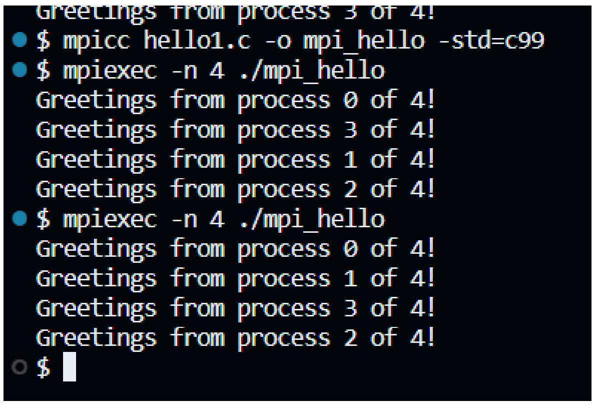


```
    } else {
        printf("Greetings from process %d of %d!\n", my_rank, comm_sz);
        for (int q = 1; q < comm_sz; q++) {
            MPI_Recv(greeting, MAX_STRING, MPI_CHAR, MPI_ANY_SOURCE,
            0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            printf("%s\n", greeting);
```

6. Explain what effect, if any, you expect this change to have.

I'm thinking this change has something to do with ordering. Before, q was keeping track of the order as it is what's used in the for loop. MPI_ANY_SOURCE like its name sounds will probably return in randomly.

7. Then try to observe whether or not this happens. Explain your method and present results.



```
  Greetings from process 3 of 4!
● $ mpicc hello1.c -o mpi_hello -std=c99
● $ mpiexec -n 4 ./mpi_hello
  Greetings from process 0 of 4!
  Greetings from process 3 of 4!
  Greetings from process 1 of 4!
  Greetings from process 2 of 4!
● $ mpiexec -n 4 ./mpi_hello
  Greetings from process 0 of 4!
  Greetings from process 1 of 4!
  Greetings from process 3 of 4!
  Greetings from process 2 of 4!
○ $
```

The processes are randomly ordered. After running two different times it is clear, the order is not 1,2,3,4 or the same order every time the program is executed. Before, we used q=1 in the for loop before calling MPI_Recv and specified q in the parameters which meant results will be displayed in the same way as the for loops outputs. Whereas, MPI_ANY_SOURCE makes it so that the order will not take in the q order from the for loop.