- Design a program using concepts of inter-process communication ordinary pipes in which one process sends a string message to a second process, and the second process counts the number of vowels in the message and sends it back to the first process

- You will write two simple programs 'my_npipe_reader.c' and 'my_npipe_writer.c' that use a named pipe to communicate. The 'my_npipe_reader' program will set up a named pipe using 'mkfifo()', open it read only, and read strings from it until it receives the string 'exit'. The writer will open the named pipe file, read strings from the user and write them to the named pipe. When the user enters 'exit', the program will write the string to the pipe and then exit. Execution should look something like this (note that you must start the reader first):

  ```
  reader:
  $ ./my_npipe_reader
  Creating named pipe: /tmp/mypipe
  Waiting for input...Got it: 'hello world'
  Waiting for input...Got it: 'foober goober'
  Waiting for input...Got it: 'exit'
  Exiting
  ```

  ```
  writer:
  $ ./my_npipe_writer
  Opening named pipe: /tmp/mypipe
  Enter Input: hello world
  Writing buffer to pipe...done
  Enter Input: foober goober
  Writing buffer to pipe...done
  Enter Input: exit
  Writing buffer to pipe...done
  Exiting
  ```

  Note that the 'my_npipe_reader' and 'my_npipe_writer' need to be executed in separate shells at the same time. The reader stops at 'Waiting for input...' until it recieves data from the pipe (the read completes).