# Homework 3

1. Functions used: fgetc() fopen() fclose() fscanf() printf()

2. Open the `/dev/urandom` "file" for reading; call it devran.c.

   - Read and display ten printable ascii characters (ie. those having codes between 32-126, inclusive)
   - Use a do-while loop to read in characters which continues until a character is in between 32-126.
   - Count the number of characters that get rejected by the do-while loop.
   - HINT: use a for loop counting from 0 to 9, and inside it have the do-while loop. Increment the rejected count for all characters read in with fgetc(); decrement the rejected count for accepted characters.
   - Report on each of the accepted characters: `"%d:%c\n"`, (use the for loop index and the character)
   - Reported on the rejected character count with `"rejected %d characters"`

3. Make a copy of the program from the previous problem; call it devranfifo.c. Modify this variant as follows:

   - This variant should read the quantity of random ascii characters it should display from a FIFO to be called devranfifo.in.
     HINT: use `fscanf(fp,"%d",&qty)` where `fp` is obtained with `fp= fopen("devranfifo.in","r");`
   - devranfifo should create the fifo *itself*. (hint: mkfifo, and use octal numbers with mode, ie. use a leading 0)
   - To exercise the program, use:

     ```
     devranfifo&
     echo 20 >devranfifo.in
     ```

   - Note how this procedure writes to the FIFO *prior* to devranfifo reading from it.

FYI: There are two random number devices:

1. `/dev/random` : this uses an "entropy pool" of truly random bits. However, you may run out of such bits and need to wait until additional environmental noise is accumulated (ie. your process will be blocked for awhile).

2. `/dev/urandom` : this uses a pseudo-random number generator; its output is not truly random and may be susceptible to a (future) cryptographic attack. However, it doesn't block and its output appears quickly.