Exercise (1) : input.c

Write a program that reads standard input and prints the characters greater than 40 (, say).

The primary purpose of this exercise is to learn how to read the standard input and use the characters read as part of the program. The universal practice to read input typed on the keyboard is as follows:

```
# include <stdio.h>
    ==> necessary to get value of EOF ( which is an int )

int c;
    ==> variable to read character
        This must be int and not char for comparison with EOF

while ( (c = getchar()) != EOF ) {
    ==> The general way. The C code within this while loop
        is executed once for each character typed on the
        keyboard.

    ==> The brackets ( ) around     c = getchar()
        are a must since "=" operator has lower precedence
        than the "!=" operator.

    ==> EOF is actually not a character.  getchar() returns
        the value of EOF ( normally -1 ) when end of input is
        indicated in the input.  This is done by typing the
        Ctrl D  character on the keyboard on a fresh line.
        This can be changed using "stty" command.
```

While testing your program, you must always check boundary conditions. In this program, characters whose ASCII values are less than 40 must also be entered while running the program

(Ex: ` ! @ # $ % ^ & *)

Exercise (2) : escape.c

Try printing the full range of escaped characters with printf :

(**\a** **\b** **\c** ... **\z**)

Explain the result on the screen. You may have problem with some of these control characters because of the way the terminal handles them.

Modify the program slightly to prove your theories (for **\b** **\r**). While testing, if the terminal hangs up (because of some special escape sequences), one way of recovery is to switch off the terminal and switch it on again.

Exercise (3) : degree.c

Write a program to convert 300, 290, 280, ... 0 degrees Fahrenheit to Celsius and print the results in a nice tabulated fashion.

Exercise (4) : squeeze.c

Copy input to output, replacing series of consecutive spaces by a single space.