In this assignment, you will develop a series of procedures which convert data from one format to another.

While we can view any data stored in a digital computer as a series of bits or bytes, the encoding applied to the data gives the data its meaning. This encoding may be applied during input/output, or the data may be manipulated in its encoded form directly by machine instructions and stored as such. Several programming languages include language features to manipulate or interpret data according to a particular format during I/O (e.g., C++), whereas others leave the encoding and translation from one encoding to another to library procedures (e.g., C).

You are asked to develop the following procedures:

```c
void int_to_ascii(char *buf, int v, int l)
```
This procedure should encode the given binary number in `v` into the buffer `buf` right-justified as an ascii encoded decimal number. The buffer can be assumed to have at least `l` bytes of space. Any bytes to the left of the number must be blank (i.e., ASCII space character, 0x20).

```c
void int_to_ascii_hex(char *buf, int v, int l)
```
This procedure should store the given binary number in `v` into the buffer `buf` right-justified as an ascii encoded hexadecimal number. The buffer can be assumed to have at least `l` bytes of space. Any bytes to the left of the hexadecimal number must be blank.

```c
void float_to_ascii(char *buf, float v, int l)
```
This procedure should encode the given binary floating point number in `v` (IEEE 754) into the buffer `buf` right-justified. The buffer can be assumed to have at least `l` bytes of space. Any bytes to the left of the number must be blank. The desired format is nnnn.mm (i.e., two digits after the decimal point, as many digits as needed to the left of the decimal point). You may assume that `l` will be sufficient to hold the encoded number.

```c
int int_to_bcd(int v)
```
This procedure should return a packed binary coded decimal representation¹ of the given binary number. Assume that a sign digit of 1100 (hex C) for positive (+) and a 1101 (hex D) for negative is stored in the lower nibble of the rightmost byte. The largest packed decimal number which can be stored into a 32 bit value is +9999999 and the smallest is -9999999.

```c
void bcd_to_ascii(char *buf, int v)
```
This procedure should store the given packed BCD number in `v` into the buffer `buf` right-justified as an ascii encoded decimal number. The buffer can be assumed to have at least 8 bytes of space. Any bytes to the left of the number must be blank.

Test your procedures by writing a main program. The skeleton is given below:

```c
write(1, "Enter a number (7 digits max) : ", 32);
scanf("%d", &v);
int_to_ascii(buf, v, 8);
```

¹https://en.wikipedia.org/wiki/Binary-coded_decimal
write(1, buf, 8);
write(1, "\n", 1);

write(1, "Enter a number (no maximum) : ", 30);
scanf("%d", &v);

int_to_ascii_hex(buf, v, 8);
write(1, buf, 8);
write(1, "\n", 1);

write(1, "Enter a float number (7 digits max) : ", 37);
scanf("%f", &f);

float_to_ascii(buf, v, 8);
write(1, buf, 8);
write(1, "\n", 1);

write(1, "Enter a number (7 digits max) : ", 32);
scanf("%d", &v);

v = int_to_bcd(v);
int_to_ascii_hex(buf, v, 8);
write(1, buf, 8);
write(1, "\n", 1);

bcd_to_ascii_hex(buf, v);
write(1, buf, 8);
write(1, "\n", 1);

Requirements:

1. Your program should not contain any standard i/o calls other than scanf.

2. Your program should not use any library functions to perform the desired functionality. You also are not permitted to use any existing code, other than what has been discussed in class.

3. If any error is detected, your program should print an appropriate error message and exit.

Submission:
Your submission should include the source code written in C called encode.c, and a Makefile. When the makefile is invoked, it should generate a binary called encode in the current directory.

Recommendations:

1. Debug as you write your code.

2. Test the program with a variety of inputs.

No hardcopy is required.