PRACTICE PROBLEMS - SET 3

#1. The following short questions have all appeared on old midterms and finals:

(a) Convert the hexadecimal number ABCD to a binary number.

(b) Consider the following four non-decimal numbers, where the subscripts indicate the base of each number: $A8_{16}$ 251_8 2222_4 10100111_2. Which number is the smallest, and which is the largest?

(c) What is the binary representation of a, where a = 27_{16} - 27_8 ?

(d) What is the octal (base 8) representation of the hexadecimal value B38?

#2. Write a "binary to decimal" converter. Write a program that asks a user for a four digit binary number. Check whether each digit is valid (0 or 1); if you find a digit that isn't, ask again for a four digit number. If all four digits are valid, then calculate the decimal equivalent and write it to the screen. For example:

> Enter a four-digit binary number: **1301**
> Enter a four-digit binary number: **1101**
> Binary 1101 equals decimal 13

#3. Write an "octal to binary" converter. Write a program that asks a user for a 1-3 digit octal number (an octal number from 0 to 777). Check whether each digit is valid (0 to 7); if you find a digit that isn't, ask again. Once you have a valid octal number, convert it to a binary number. For example:

> Enter an octal number (0 – 777): **186**
> Enter an octal number (0 – 777): **263**
> Octal 263 equals binary 010110011

#4. Very simply, ask a user to type whatever they want, then respond by telling them how many keystrokes they used. For example:

> Type whatever you want: Here's whatever I want
>
> You typed 23 keystrokes (including the return/enter key)

#5. Something a little more complicated. Write a program that asks a user to enter a word, and then decides if the word is in alphabetical order. For example:

```
Enter a word:  Beer
The word is in alphabetical order
```

or

```
Enter a word:  SOS
The word is NOT in alphabetical order
```

Note that to compare letters, you'll need to convert all letters to one case. And if the user enters something other than a letter, the program should respond appropriately. For example:

```
Enter a word:  Gr8
That's not a word!
```

#6. Write a program that asks a user to enter a sentence (letters, numbers, white space, punctuation, …), and that then counts the total number of characters, as well as the number of letters and numbers. For example:

```
Enter a sentence:  1 potato, 2 potato, 3 potato, 4
31 characters including 18 letters and 4 numbers
```

You'll need to repeatedly use the getchar() function until you read a newline character ('\n'). You can determine whether a character is a letter with the function isalpha(), and a number with the function isdigit(). Both of these require that you #include <ctype.h>. You send these functions a char, and they return a 0 or 1, depending on whether the argument has a particular property. For example:

```
char c = 'A';

if (isalpha(c))
  printf ("is a letter");
else
  printf ("is not a letter");
```

will yield:

```
is a letter
```

#7. On BB I've posted a file grades.txt that contains a list of integer grades. Write a program that opens that file, reads all of the grades, and writes the following information to the screeen: the total number of grades, the number of grades above and below 50, and the average grade.

#8. Another quick question. Write a program that creates a copy of grades.txt, by reading each grade from the original file, and then immediately writing that value to a file of a different name.

#9. Open the file onetwo.txt with a simple text editor (e.g. Notepad on Windows, or TextEdit on a Mac), and you'll discover a sentence, but with each word on a separate line; something like:

```
One,
two,
buckle
my
shoe.
```

It's important to realize that although you can't see them, the file contains newline characters at the end of each line. Write a program that reads the input file one character at a time, and writes the words to the screen, all on one line; for example:

```
One, two, buckle my shoe.
```

#10. Write a function F2C that corresponds to the following declaration:

```
double F2C (double tempF);
```

The argument tempF is a Fahrenheit temperature; the function returns the equivalent temperature in degrees Celsius, according to the following formula:

```
tempC = 5/9 * (tempF-32)
```

Then write a main ( ) function to ask a user for a Fahrenheit temperature, use F2C to convert that to Celsius, and output that value to the screen.

#11. Write a function round that corresponds to the following declaration:

```
int round (double a);
```

The argument a can be any floating point number, positive or negative; the function rounds that value to the nearest whole number, and returns that value as an int. If you then write a main ( ) function to accompany round, you could then have input and output like this:

```
enter a number:  12.3
rounded to the nearest whole number:  12
```

or

```
enter a number:  -14.7
rounded to the nearest whole number:  -15
```

#12. Write a function num_digits(n) that returns the number of digits in a positive integer n. If n <= 0, the function should return -1. The function main ( ) used to call the function would look like this:

```
#include <stdio.h>

int num_digits(int n);  /* function declaration */

int main( ) {

   int num, digits;

   printf("enter a positive integer:  ");
   scanf("%d", &num);

   digits = num_digits(num);  /* function call */

   if (digits == -1)
      printf ("\nsorry; you entered a non-positive number\n\n");
   else
      printf ("\nthe number of digits in %d is %d\n\n", num, digits);

   return 0;
}
```

... this is where you write the function num_digits