Extra Problems for Weeks 3 and 4

#1. Write a program that takes the role of a store clerk. Ask a user to enter two floating point numbers: the cost of an item, and the amount of money remitted to pay for the item. Then respond appropriately: calculate the change due to the customer, or ask the customer for more money. For example:

```
Cost of the item:  3.56
Amount tendered:  5.00
Change:  1.44
```

or

```
Cost of the item:  3.56
Amount tendered:  3.00
Still due:  0.56
```

#2. Another one that's not too hard. From the text, write a program that determines the number of digits (from one to four) in a positive number:

```
Enter a number:  374
The number 374 has 3 digits
```

If the number entered is negative, or has more than four digits, print an appropriate response. Otherwise, determine the number of digits, and respond as shown.

#3. Write a program that converts a time specified as am/pm into a military time (00:00 to 23:59). For example:

```
Enter a time (hr min):  7 25
am (1) or pm (2):  1
Military time:  07:25
```

or

```
Enter a time (hr min):  12 30
am (1) or pm (2):  1
Military time:  00:30
```

or

```
Enter a time (hr min):  1 30
am (1) or pm (2):  2
Military time:  13:30
```

To specify that an integer print with a leading zero (e.g. 07), print it as %.2d.

#4. This one's a bit more complicated. Write a program to calculate a babysitting charge. Imagine that a babysitter is willing to work anytime between 6:00 pm and 5:59 am. Ask the babysitter to enter a start and an end time, each as two integers (see below). Do not ask the user to specify am or pm. Check that the times entered are valid: 1 <= hours <= 12, 0 <= mins <= 59. Check that the start time precedes the end time. Then calculate the babysitting charge at a rate of $4.50 / hour, calculated to the minute and then rounded up to the nearest dollar.

```
Enter a start time:  7 20
Enter an end time: 12 45
The babysitting charge is $25
```

There are surely many ways to do this. But if you're stuck, try devising a way to convert each of the two times into minutes since 6:00 pm, which is relatively easy for times until 12:59 am, and only a little more difficult for times beginning at 1:00 am.

#5. We turn our attention to loops, and we begin with a couple of questions that are relatively straightforward.

Write a program that calculates the sum of odd numbers from 1 to N:

```
1 + 3 + 5 + 7 + 9 + ... + N
```

where N is provided by the user. Make sure to check that the user enters an N that is odd and greater than zero.

#6. The following is a series representation for pi:

```
pi = 4 - 4/3 + 4/5 - 4/7 + 4/9 - ...
```

Write a program that asks a user to input a (positive) number of terms, and then calculate the approximation to pi by evaluating that many terms of the series.

#7. Write a program that calculates the number of digits of any positive integer.

#8. Write a program that repeatedly asks a user for integers, UNTIL the user enters a zero. Calculate the number of positive and negative integers that were entered. For example:

```
Enter an integer:  2
Enter an integer:  14
Enter an integer:  -3
Enter an integer:  0

You entered 2 positive and 1 negative values.
```

#9. The following is known as the Fibonacci series of numbers:

0 1 1 2 3 5 8 13 21 34 55 89 ...

The first Fibonacci number is 0, the second Fibonacci number is 1. After the first two, each successive number can be calculated by summing the previous two.

Write a program that asks a user for a number of terms, and then prints that many terms of the series to the screen.

#10. Write a C program that evaluates $\tanh^{-1}(x)$ from the series

$$\tanh^{-1}(x) = x + x^3/3 + x^5/5 + x^7/7 + x^9/9 + \ldots$$

where $x < 1.0$ and $x > -1.0$. The program should begin by asking a user to enter x, and the number of terms to sum, N. If x is not between -1.0 and 1.0 or if N is less than 5, the program should repeatedly ask for input until these conditions are met. When the input is valid, the program should estimate the value of $\tanh^{-1}(x)$ by summing the first N terms of the series, and then print the value of $\tanh^{-1}(x)$. For example:

Enter x and the number of terms of series: 0.4 3
Enter x and the number of terms of series: 1.5 11
Enter x and the number of terms of series: 0.5 7

tanh-1(0.500000) = 0.549303531

#11. The following series is an approximation to sin(x):

$$\sin(x) = x - x^3/3! + x^5/5! - x^7/7! + \ldots$$

Ask a user for a number of terms and for a value of x (specified in degrees), and then calculate the approximate value of sin(x), and indicate how much this differs from the "exact" value that you calculate with the sin function in math.h. For example:

how many terms? 5
value of x? 45.
approximate value of sin(45.000000) is 0.707107
this is 1.750320e-009 greater than the exact value

#12. Write a program that prints a one-month calendar. The user specifies the number of days in the month and the day of the week on which the month begins:

Enter number of days in month: 31
Enter starting day (1=Sun, ..., 7=Sat): 3

and the code produces the following:

```
          1   2   3   4   5
 6   7   8   9  10  11  12
13  14  15  16  17  18  19
20  21  22  23  24  25  26
27  28  29  30  31
```

This program isn't as hard as it looks. The most important part is a for statement that uses a variable i to count from 1 to n, where n is the number of days in the month, printing each value of i. Inside the loop, an if statement tests whether i is the last day in a week; if so, it prints a newline character.

#13. Back to the Fibonacci series of numbers:

0 1 1 2 3 5 8 13 21 34 55 89 ...

The first Fibonacci number is 0, the second Fibonacci number is 1. After the first two, each successive number can be calculated by summing the previous two.

Write a program that does the following:

1. asks a user for two positive integers n1 and n2;
2. checks that n1 and n2 are greater than zero, and that n2 > n1; if the numbers aren't valid, the program should repeatedly ask the user for valid n1 and n2, until they're entered;
3. prints the values of the Fibonacci numbers between n1 and n2; for example, if n1 = 7 and n2 = 10, then:

Fibonacci numbers 7 through 10 are 8 13 21 34.

#14. Write a program that repeatedly asks a user for an integer until the user enters a value that is less than the previous value. At that point, calculate the average increment in the set of increasing values. For example

enter a number: -14
enter a number: 3
enter a number: 11
enter a number: 11
enter a number: 18
enter a number: 13

average increment is 32/4 = 8.00

Generalize the code to look at the first two numbers entered, and decide whether the sequence is increasing or decreasing. Continue to accept values until the sequence is disrupted.

#15. Imagine that you'd like to calculate the total play length of a music CD, the sum of the play lengths of each of the songs.

Write a program that repeatedly asks a user to input the play length of a song, in the form min:sec (e.g. 4:37). The appropriate format specifier is "%d:%d". Check that the number of seconds is less than 60. To indicate the end of the list, the user would enter a negative value for min.

Count the number of songs, calculate the total play length, and output the results.

For example, input/output might look like this:

Length of song (min:sec) : 4:14
Length of song (min:sec) : 3:99
Invalid entry
Length of song (min:sec) : 3:39
Length of song (min:sec) : 17:32
Length of song (min:sec) : 49:13
Length of song (min:sec) : -1:00

The total play length of 4 songs is (hh:mm:ss) 1:14:38.

#16. Consider the following sequence of numbers. Notice that from the first to the second you add one, from the second to third you subtract three. That behaviour is then repeated over and over again; for example:

7, 8, 5, 6, 3, 4, 1, 2, -1, 0

Write a program that asks a user for the first three integers in a sequence, and for a number of terms N. Then print the sequence, beginning with the three integers the user entered, followed by N-3 other terms that follow the pattern defined by the first three integers. For example:

Enter three integers: -4 3 4
How many terms: 8
The sequence is -4 3 4 11 12 19 20 27

#17. Imagine that rather than using coins valued at 1, 5, 10, and 25 cents, that we use coins valued at 1, 3, 5, and 7 cents. Write a program that asks a user for a positive integer, and then calculates the number of each coin required to make up that amount. For example:

Total amount: 69

9 7-cent coins = 63
1 5-cent coins = 68
0 3-cent coins = 68
1 1-cent coins = 69

Write this using a loop, rather than calculating each of the four numbers separately.

#18. From an old midterm (where the question specified that this be written as a separate function). Here, write a program with only a main ( ) function. Consider a triangle of n rows; for example:

for n = 2:

```
 1
23
```

for n = 5:

```
    1
   23
  345
 4567
56789
```

and for n = 9:

```
        1
       23
      345
     4567
    56789
   678901
  7890123
 89012345
901234567
```

Write a program that asks a user for a number between 1 and 9, and then prints the corresponding triangle to the screen. Recognize that you'll need a pair of nested loops, one for each line of the triangle, and one to generate the numbers on a particular line. And if it seems hard to write the spaces to the left of the numbers, begin by writing the output without the leading spaces.

#19. Write a program to play a number guessing game with a user. You can generate a random integer between 1 and 100 with the following lines of code:

```
#include <stdlib.h>
#include <time.h>

...

int number;

srand ((unsigned) time(NULL));
number = rand()%100+1;
```

Then write the code to play the following game:

```
Guess a number (1-100)    ::  50
Higher than 50; guess again        ::  75
Lower than 75; guess again         ::  65
Higher than 65; guess again        ::  70
Higher than 70; guess again        ::  73
Good guess - you win!
```

where you check that each entry is between 1 and 100. Once that's working, you may want to consider adding messages that tell a user when they're not playing the game logically; for example:

```
Guess a number (1-100)    ::  50
Higher than 50; guess again        ::  75
Lower than 75; guess again         ::  45
THINK! - I already told you that the number is > 50
Try again  ::  65
...
```