Lab 3: Formatting and flow control

Goals

Understand how to use format controls to create tabular output.
Understand how to execute a basic block iteratively.

Background

Review

Arithmetic:

- Reading integer values from the keyboard:

```
int x;         // receives input value
int howmany;  // howmany values were read succesfully
howmany = fscanf(stdin “%d”, &x);
x = x + 5;
```

- Printing integer values to the screen:

```
fprintf(stdout, “The value of x is: %d\n”, x);
```

- Printing *x* first in decimal and then in hexadecimal *on the same line* with specified field widths: (To catenate the output of multiple fprintfs() on the same line, *omit the \n character* from all but the final fprintf().)

```
fprintf(stdout, “X in decimal: %8d”, x);
fprintf(stdout, “X in hexadecimal: %6x\n”, x);
```

Iterative execution

In virtually all useful programs it is necessary to repetitively execute a statement or block of statements *while* or *until* some condition *remains* or *becomes true* or *false*.

Because of all of the possible formulations of while/until, remains/becomes, and true/false it is always the case that there are always *multiple* correct ways to construct such iterations. And, alas, there are *even more incorrect ways* to construct them.

For this lab we will focus on *while* some condition *remains true.* This structure is commonly called the "*while loop.*"

- Structure:

  ```
  while (condition)
  {
       one-or-more statements;
  }

  or
  while (condition)
      exactly-one-statement;
  ```

- Important notes

  - the statement or statements controlled by the *while()* are called the *body of the loop.*

  - It is necessary that the *body of the loop modify the value of the condition*.. Why?

  - Proper use of indentation is *critical* for human readability

  - But indentation is *completely irrelevant* to the C compiler.

- Example: Print the integer values between 10 and 1 in decreasing order. (Note: This is *not* a complete C program. It needs to include *stdio.h* and it needs a proper main() function.

```
int value;
value = 10;

fprintf(stdout, " Table of values \n");

while (value > 0)
{
 fprintf (stdout, "%3d\n", value);
 value = value - 1;
}
```

- Note we only want the heading to be printed *one time.* So that *fprintf()* must be placed *outside the loop*.

  Thought experiment: What does the following code segment do:

  - Refuse to compile because of missing {} ?
  - Compile fine and work fine?
  - Compile fine but work not so fine?; if so how would it work not so fine??

```
int value;
value = 10;

fprintf(stdout, " Table of values \n");

while (value > 0)
   fprintf (stdout, "%3d\n", value);
   value = value - 1;
```

Assignment:

Part I:

Complete exercises 10076, 10077, and 10078 in the
Original -> iteration -> language_support -> while
section of Codelab. In exercise 10077 (and possibly others), it is not necessary to use both of the variables they provide you, but it is perfectly OK to do so.

Part II:

Write a program called lab3.c that does the following:

- Read an integer value *x* from the standard input.
  Using a *while loop* compute X^n for the n = 1, 2, .. 10 and print a table of the following format containing the values *n* and X^n in both *decimal* and *hexadecimal* representation. *Do not use any C language facilities or C library functions (e.g. pow()) that were not discussed in this or previous lab documents!*
- You will need three int variables in this program. I would call them *n*, *x*, and *xn* (X^n).
- Which ones need to change inside the loop? In what way do they change?
- Your output should look EXACTLY like this table if (and only if) the input value is 2.

```
      Powers of 2 <---- print value of x (not the constant 2)
 1            2        2
 2            4        4
 3            8        8
 4           16       10
 5           32       20
 6           64       40
 7          128       80
 8          256      100
 9          512      200
10         1024      400
```

- To ensure that your output remains properly aligned, your first column should be 3 digits wide, the second 12, and the third 8. You should experimentially position the heading as shown.

- Experimentally determine what is the largest input value that produces mathematically correct output. Add a comment to your program warning potential users that *it will not work for values larger than the one you identify.*

Turn In Work

Show your TA that you completed the assignment. Then turn in your lab3.c program using the command:

sendlab.101.*section_number* 3 lab3.c