Assignment 2

Due Saturday, July 20th, 2013

This assignment is worth 6% of your grade.

IMPORTANT:
The only libraries that can be used are stdio.h and math.h. For those already familiar with arrays and strings, you are *not allowed* to use them to solve this assignment.

Submit your code on cuLearn in ONE file called a2.c

Inside of the `main` function you should separate parts belonging to different questions as before by:

```
 /*
==================================================
              question X
==================================================
  */
```

Outside of the `main` you should separate functions belonging to different questions by:

```
 /*
==================================================
              Function(s) for question X
==================================================
  */
```

(Replace X with the number of the question you are answering.)

As always, your code must compile, otherwise the assignment will be graded with mark 0. For that reason, if you run out of time and/or one of your answers contains code that does not compile, then comment out that section of code.

Questions:

1. Write a program that tests if a given horizontal segment intersects a given vertical segment (definition: two segments *intersect* if they have a point in common). Specifically, the program should prompt the user to enter 3 numbers for the horizontal segment: the y-coordinate of the horizontal segment, the x-coordinate of the left endpoint and the x-coordinate of the right endpoint. Then the program should prompt the user to enter 3 numbers for the vertical segment: the x-coordinate of the vertical segment, the y-coordinate of the bottom endpoint and the y-coordinate of the top endpoint. The program should print "yes, they intersect" if the two segments intersect and "no, they do not intersect" if they do not intersect.

   Example:
   Your program:
   `Enter the 3 numbers for horizontal segment:`
   User: `0.0 -1.0 1.0`
   Your program:
   `Enter the 3 numbers for vertical segment:`
   User: `0.5 -2.0 2.0`
   Your program:
   `yes, they intersect`

Another Example:
Your program:
```
Enter the 3 numbers for horizontal segment:
```
User: `3.0 1.0 5.0`
Your program:
```
Enter the 3 numbers for vertical segment:
```
User: `0.5 1.0 2.0`
Your program:
```
no, they do not intersect
```

2. Write a program that prompts the user to enter a positive integer n and prints the value equal to $n*(n-1)*(n-2)*\cdots*2*1$.

3. Write a program that has a function called `iz_zero`. The function should take as input a number, x, and return the character 'z' if x is equal to zero and character 'n' if x is not equal to zero.

   In the *main*: your program should:
   1. Prompt the user to enter a number,
   2. Make a call to function `is_zero` where you pass that number to the function
   3. Print the character that the function returned.
   4. Test what the function returned and based on that test either print:
   ```
   User entered zero
   ```
   or
   ```
   User entered a non-zero number
   ```

4. Write a program that has a function called `my_rectangle`. The function should take an as input a non-negative integer n and two characters ch1 and ch2 (thus your function should have 3 arguments). You function should draw a rectangle where horizontal lines are drawn by printing the ch1 6 times and where the vertical lines are drawn by printing n times the following: ch2 followed by spaces followed by ch2 (see example below).

   In the *main*: your program should:
   1. Prompt the user to enter a non-negative integer followed by a space followed by a character followed by a space followed by another character.
   2. Repeat step one as long as the user is entering a negative number.
   3. Make a call to function `my_rectangle` where you pass the three values to the function.

   Example:
   Your program:
   ```
   Enter a non-negative integer followed by a space
   followed by a character followed by a space followed by another character
   ```
   User: `-100 C b`
   Your program:
   ```
   Enter a non-negative integer followed by a space
   followed by a character followed by a space followed by another character
   ```
   User: `3 A b`
   Your program:
   ```
   AAAAAA
   b    b
   b    b
   b    b
   AAAAAA
   ```

5. Write a program that has a function called `average` That function should takes as input a positive integer, n. The function should, n times, prompt the user to enter a number. The function should return the average value of the entered numbers.

   In the *main*: your program should:
   1. Prompt the user to enter a positive integer,
   2. Make a call to function `average` where you pass that value as an input
   3. Print the value that the function returns.

   Example:
   Your program: `Enter a non-negative integer`
   User: `3`
   Your program: `Enter a number`
   User: `4.5`
   Your program: `Enter a number`
   User: `-4.5`
   Your program: `Enter a number`
   User: `9.0`
   Your program:
   `The average of the entered numbers is 3.0`

6. Repeat the previous question except this time your function should be called `min` and it should return the minimum of all the entered characters.

7. Repeat the previous question except this time your function should be called `standard_dev` and it should return the standard deviation of the entered numbers. The expression for standard deviation is:

$$standard\_deviation = \sqrt{\frac{sum\_squares}{n} - average^2}$$

   where $sum\_squares$ is the sum of the squares of the entered numbers, and where $average$ is the average of the entered numbers. Note that you can create this function by modifying function `average` that you were asked to write earlier.

8. Write a program that has a function called `invert_caps`. The function should take as input a letter and if that letter is an upper case letter it returns the same letter but as a lower case, otherwise, if that letter is a lower case letter your function should return the same letter but as an upper case.

   In the *main*: your program should:
   1. Prompt the user to enter a letter,
   2. Make a call to function `invert_caps` where you passed that value as an input
   3. Print the letter that the function returns.

   Hint: To answer this question you should recall data type "char" and its connection to ASCII Code (in Chapter 2 of your book – pages 57 and 58). You should also recall how to cast from one data type to another.

9. The value for π can be determined by the series equation

$$\pi = 4\Big(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \dots\Big)$$

   Write a program to approximate the value of π using the formula given including terms up through $\frac{1}{99}$.

10. Write a program that determines the day number (1 to 366) in a year for a date that is provided as input data. As an example, January 1, 1994, is day 1. December 31, 1993, is day 365. December 31, 1996, is day 366, since 1996 is a leap year. A year is a leap year if it is divisible by four, except that any year divisible by 100 is a leap year only if it is divisible by 400. Your program should accept the month, day, and year as integers. Include a function `leap` that returns 1 if called with a leap year, 0 otherwise.