Section Exercises 1: C basics

Each week we will try to hand out a set of questions and exercises to go along with the material covered in lectures. These exercises will give you more practice with the concepts introduced in class and help you work up to the programming assignments and prepare for exams. Usually we'll hand out the exercises in Monday's lecture and the solutions in section. You are encouraged to look at these exercises before section. Some of the more complex problems may be worked through in section and you should feel free to bring questions to section about any of these exercises. **The section exercises are not to be handed in or graded.**

Problem 1: Warm-up questions on style

a) When do you need to use constants?

b) What makes a good variable or function name?

c) When is it a good idea to decompose a section of code into its own function call?

d) Is creating a function that has just one of line of code a good idea or a practice to avoid?

Problem 2: Arithmetic expressions

Pay attention to precedence rules while evaluating the following arithmetic expressions:

a)

```
(1 - (2 + 8 / 6) * 4 % 3 * 1) / 2;
```

b)

```
int num1, num2;
double num3, num4;

num1 = 5.999;
num2 = -2;
num3 = 5;
num4 = num1 / num2++ + (int) (num3 * (num3 / 2));
```

What is the value of **num4**?

Problem 3: Boolean expressions

a) To what will the following Boolean expression evaluate? (`TRUE` or `FALSE`) Assume `a` and `b` are `int`s.

```
((a != a) || ((a == b) && (a != b))
```

b) What about this one?

```
(a == a) || (a != a)
```

c) The following Boolean expression needs no parenthesis, but is **much clearer** when written with them. Insert the parenthesis to indicate how the compiler will read this line, which should be exactly how you would write this line if it were in one of your programs:

```
y % 4 == 0 && y % 100 != 0 || y % 400 == 0
```

d) What does the following function print out?

```
int x, y, z;

x = 7;
y = 8;
z = 2;

if ((z > x) && (z + x / y - y * z == x))
{  printf("The first expression evaluates to TRUE.\n");
} else if ((y > z) || (y % z > x * z) {
   printf("The second expression evaluates to TRUE.\n");
} else {
   printf("Neither expression is true.");
}
```

Problem 4 : Divisors

Write a simple C program that takes an integer from the user and lists all the pairs of numbers that, when multiplied together, form the input number. For example,

```
Please enter a number: 32
1 * 32 = 32
2 * 16 = 32
4 * 8 = 32
8 * 4 = 32
16 * 2 = 32
32 * 1 = 32
```

Problem 5 : XOR

C does not have n operator that computes the "exclusive-or" of two booleans. "Exclusive-or" differs from "or" in that the resulting expression is true if one and only one of the two expressions is true and false otherwise. Write the function

```
bool XOR (bool x, bool y)
```

that will correctly return the XOR of the boolean values x and y.

Problem 6: Graphics

(from a previous 106A exam) You are to implement the **`DrawSwirl`** function that draws a circular pattern of evenly-spaced semi-circular arcs emerging from a center point. The function takes five parameters: the x and y of the swirl's center, the number of spokes, the radius of the swirl, and a boolean indicating whether the arcs trace clockwise or counter-clockwise from the center. Here are some example swirls and their parameters:

radius = 0.5
numSpokes = 4
clockwise = TRUE

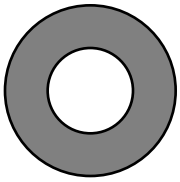
radius = 0.5
numSpokes = 8
clockwise = FALSE

radius = 0.35
numSpokes = 11
clockwise = TRUE

```
/*
 * Function: DrawSwirl
 * Usage: DrawSwirl(1.0, 1.0, 4, 0.5, TRUE);
 * ----------------------------------------
 * Draws a figure centered at (cx, cy) consisting of numSpokes half-
 * circle arc segments evenly spaced around a circle. The radius
 * is the distance from the center to the end of each arc. The
 * clockwise parameter controls whether the arcs trace clockwise or
 * counter-clockwise.
 */
void DrawSwirl(double cx, double cy, int numSpokes, double radius,
               bool clockwise)
```

Problem 7: Random graphics

To give you practice using the extended graphics and random libraries, you're going to write a program to draw some donuts. While the drawing itself is fairly simple, you're going to need to think a little bit about how to decompose it. To draw a donut, you draw an outer circle of a random density, and then an inner circle that is filled in with white.



Your program will draw donuts at random places on the screen until the user clicks the mouse to stop the donut parade. To brush up on your math skills, we want you to make sure that each of the donuts entirely fits on the screen.

To figure out how to do the drawing, look at extgraph.h and graphics.h You will also need to use the functions from random.h in order to choose the values for random placement.