

CSE 130 Fall 2009

Programming Assignment 4

Due Date: 11am, Tuesday Nov 3, 2009 Use `~cse130/submit` command from your sparky account. Do not forget to change to your working directory first. Include your name and ID number as a comment. Make sure that it runs on sparky. Write comments in your code.

Outline: Write a C program for drawing **inverted** triangles using ASCII characters as shown below. First read in the size/base, and then draw/print using several functions. You **must follow** the implementation approach described below.

Triangles: Figure shows triangles printed for different sizes. Corners of a triangle are printed using 'O'. A triangle has 3 parts; the top-line, two sides, and the bottom point. The top-line (actually the base of an inverted triangle) consists of 2 O's and an alternating sequence of a space and an 'x' in between. Let size of a triangle be n . The top-line has $n+1$ spaces and n x's between two O's. Sides of a triangle are drawn by printing two x's on each line with some spaces in between. Since the size is n , there will be n lines with 2 x's on them. The bottom point consists of just one 'O'. Also, this 'O' is printed right below the center of the top line.

```
size = 1      size = 2      size = 3
O x O        O x x O      O x x x O
 x x         x  x         x   x
  O          x  x         x   x
              O          x  x
                  O
```

Placement: First determine how to place your triangle. Note that a triangle will be printed line by line, and each line is printed from left to right. Make sure that you print `\n` after each line of a triangle. There are some initial/leading spaces that you need to print on every line (except top). You must increase these initial blanks/spaces because a triangle will become narrower and narrower (skinny) as you move from top to bottom.

Technique: To begin, print the top-line of a triangle. No initial spaces are to be printed when you print the top-line. On the second line, first print 1 space. Then on the same line print two x's with $2n-1$ spaces between them. Here n is size of the triangle. On the third line, first print 2 initial spaces, followed by 2 x's. But these two x's will have $2n-3$ spaces in between them and so on. Observe that for every line, spaces between two x's decrease by 2 (from the number spaces on the previous line).

Note that we will have n lines with 2 x's on each. Initial spaces for these n lines will increase from 1 to n . Spaces between two x's will start with $(2n-1)$, and decrease to $(2n-3)$, $(2n-5)$, $(2n-7)$, ... and so on. For the bottom, print $n+1$ spaces/blanks before 'O'.

Functions: In addition to `main()`, you **MUST** write a function `drawTriangle` and 4 line-functions. These are called line functions, as each one prints a line. You may write additional functions if you need.

(1) A function **`drawTriangle(int size)`**, which controls the whole drawing process and returns a total count of x's and O's (that have been printed) to the main program. An integer argument to this function is size of a triangle. This function calls various line functions and gets a count of O's and x's from each. It accumulates sum of these counts returned by line functions.

This function starts with a call to `topLine` for printing the top (actually the base of an inverted triangle). Update count as shown. `count = count + topLine(n);` Do not forget to skip a line in `drawTriangle` after this call.

Next, in a loop, it calls function `twoX` repeatedly to print sides of the triangle. Update count, when you get a return value for number of x's from `twoX`. (In this case, it is going to be always 2.) Do not forget to print `\n` after `twoX`. Also, inside the loop, before `twoX`, call `initialBlanks` to skip correct number of initial spaces. This loop is to be done `size` (that is `n`) times.

After sides of a triangle are printed, call function `bottom` to print just the 'O'. But before that you need to skip `(n+1)` initial blanks by calling `initialBlanks`. Skip a line after the bottom has been printed. Update count. At the end, return count to main. This number is the total of O's and x's printed by line functions.

Line functions:

(2) A function **`topLine(int n)`** to print the top of a triangle. Here `n` is the size. It prints two O's, `n` x's and `n+1` blanks as shown in the figure. It returns sum of O's and x's printed to `drawTriangle`.

(3) A function **`initialBlanks(int m)`** to print `m` initial blanks. This function is just a loop. It does not return anything.

(4) A function **`twoX(int k)`** that prints two x's on a line with `k` blanks in between. It returns number of x's printed. Count is always 2.

(5) A function **`bottom()`** to print just the 'O'. It has no input parameter, but does return number of O's printed. In this case, it case it is always 1.

Calling sequence: From `main()` call the function `drawTriangle`, and pass the size as a parameter. From `drawTriangle`, call line-functions to print the triangle. The line-functions are `topLine()`, `initialBlanks()`, `twoX()`, and the `bottom()`. You need to pass the appropriate parameters to these line-functions.

Input: First display 'Type in triangle size: '. Read the size using `scanf`. We will not test your program with a size more than 10.

Output: From `main()` call `drawTriangle`. Line functions print triangle as shown. When you return from `drawTriangle` to main, return a count of O's and x's printed by line functions. Skip a line and print this count. **Print it from main() and not from drawTriangle.**

Termination: Do not stop your program after the current triangle has been printed. Instead, go back to the beginning and ask for another size. This helps TA test your program with different sizes. If size is zero (0), then stop your program. To accomplish this, have most of your program inside a loop. See the structure given at the end. Make sure you initialize all variables again. Also print the message asking for another size. So TA knows that your program is expecting another input. Terminate your program, when size typed in is zero.

Program Structure:

```
#include <stdio.h>

// Code for function topLine
// Code for function initialBlanks

// Code for function twoX
// Code for function bottom

// Code for function drawTriangle

// Function main begins here.

int main ()
{
    int size, count;

    printf("Type in Triangle size: ");
    scanf ("%d", &size);

    while (size != 0)
    { count = drawTriangle(size);
      printf ("Total number of O's and x's is %d.\n", count);
      printf("Type in Triangle size: ");
      scanf ("%d", &size);
    }
}
```

Example Session on sparky:

Type in triangle size: 2

```
O x x O
x  x
x  x
O
```

Total number of O's and x's is 9.

Type in triangle size:

.... program continues