# CSCI 1300
# Assignment 2: Picture Interpreter

*Assigned: Thursday Sep 18, 2003*
*Due: Tuesday Sep 30, 2003*

## Honor Code Reminder

Please adhere to the policy presented on the first day of class: All work you submit to be graded must be entirely your own. Breaking this rule will result in an F grade for the entire course. You may ask friends and family for help in understanding the syntax of the C++ language, for working through examples in the text, and for understanding compliation and run-time errors, but they may not write or help you write the code for any homework assignment. If you are stuck, see the professor, TAs, or undergraduate gurus.

## The Assignment

The assignment is presented in two parts. Part 2 is an elaboration on part 1. You may find it helpful to start with part 1, and once you have it working add on to your solution to complete part 2. You should hand in only part 2 (which incorporates part 1); however, if you cannot complete part 2, hand in your part 1 solution.

### *Part 1*

Your task is to write a program that reads a sequence of numbers input by the user and interprets this sequence as commands to draw simple pictures. You can think of the sequence of numbers as a compressed encoding of a picture, and your program interprets these numbers in order to reconstruct the picture.

The input is pairs of *counts.* The first count in a pair specifies the number of asterisks to print, and the second count specifies the number of blank spaces to print. Thus, an input like

```
4 5 3 1 2 0
```

would produce the output

```
****     *** *
```

In addition, a pair whose first count is –1 indicates an end of line, and a pair whose first value is –2 indicates an end of sequence (i.e., the program should terminate). Thus, an input like

```
10 0 -1 0 2 2 2 2 2 0 -1 0 10 0 -1 0 -2 0
```

would produce the output

```
**********
**  **  **
**********
```

### *Part 2*

The pairs beginning with –1 and –2 are special commands (to end a line or end the program). In part 2, you will implement four additional special commands.

(1) The pair (–3, *n*) is a command to change the fill character from asterisk to the character whose ASCII code is *n*. A table of ASCII codes is presented in Appendix 3 of the text.

(2) The pair (–4, *n*) is a command to draw a filled square with dimensions *n* x *n*, using the current fill character.

(3) The pair (–5, *n*) is a command to draw a filled upright right triangle with height *n* and base *n*.

(4) The pair (–6, *n*) is a command to draw a filled inverted right triangle with height *n* and base *n*.

For example, the input sequence

```
-3 92 -5 3 -3 45 -4 4 -3 47 -6 3 -1 0 -2 0
```

would produce the output

```
\
\\
\\\
----
----
----
----
///
//
/
```

Of course, your program should handle a mixture of the regular and special commands. You do not need to worry that the user will input nonnumeric characters, but you should print an error and terminate the program if the user enters an invalid pair, e.g., (–10, 2) or (3, –1). Note that zero is a legal value for either or both of the counts, and a pair like (–1, –1) is valid.

### *Input*

Ideally, I would have liked to ask you to read the input sequence from a file, because if you are typing input while your program is printing output, your input will mess up the output formatting. Unfortunately, we haven't yet learned how to read input from a file. One way to solve this problem is to type the entire sequence of numbers on one line and then hit "enter". The program will begin processing this stream only after you hit "enter". A second solution which will be useful to you in testing your program is to put a sequence of numbers in a file (e.g., call the file assign2.dat), and then use the *input redirection* feature of the Windows Command Prompt (this also works in UNIX/LINUX). If your compiled program is called assign2.exe, then you can type the following at the command prompt:

```
assign2 < assign2.dat
```

This command will cause input from the file assign2.dat to be treated as characters the user is typing. That is, it *redirects* input from the keyboard to input from the file.

Because everyone in the class must make their program work

## Handing in Your Program

Follow these instructions of your program will not be graded.

(1) By 11:55 p.m. of the assignment due date, you must submit your program via the WebCT system. In a browser window, go to `webct.colorado.edu` and follow the directions to sign on. Click on "submit assignments" and "homework 2", and upload your C++ source code (the file with C++ instructions).

(2) The first line of your program should be a comment with the assignment #, your identikey ID (also your webct ID) and name, e.g.,

```
// Assignment 2    IDENTIKEY mozer      Michael C. Mozer
```

(3) If you have any sort of note to your TA (e.g., you weren't able to get some part of the program to work), write it in comments below the first line above. Your program should contain comments describing the operation of the program, and you should comment each block of code to explain at a high-level what the different parts of your program are doing. Comments should not simply paraphrase the C++ but give the reader and overview of what the code does.

(4) Late assignments will not be accepted.

(5) Come to recitation on the Wednesday following the due date with a copy of your program, either on floppy disk, or downloadable via email. You will be asked to demonstrate the program for your TA. You should make up a sexy example input that shows off your program.