Homework 4
Assigned in Laboratory 6
Due Start of Laboratory 8

Perform the following activities. You may work alone or in pairs —pairs must be from the same lab section. Although you are allowed to talk with people outside of your group (alone or pair) regarding assignment requirements and debugging, the work of each group must be its own, and pledged accordingly.

Assignment files

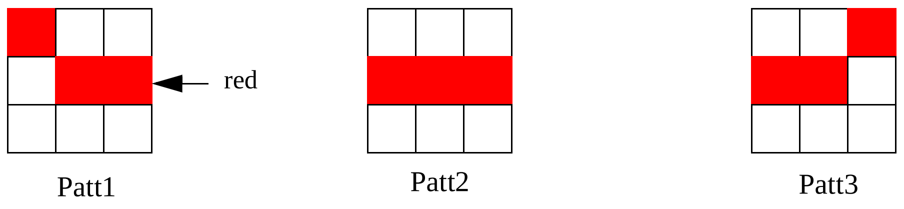
To assist you, we have created: a Visual C++ workspace file `hw04.dsw`, a Visual C++ project file `hw04.dsp` and a C++ source file `hw04.cpp`. The files can be used in the Stacks. If you work elsewhere you will need to have EzWindows installed and will need to update the project settings to reflect where the EzWindows header and library files are located. Remember class personnel will not assist you in configuring your personal system. The source file has comments and some code that will assist you in your program development. Additional information on EzWindows can be found in Appendix E of the text. The three files have been placed in a self-extracting file `hw04files.exe` at the class website.

Objective

Gain experience with graphical programs, conditional and looping constructs, data validation, and functions. Program quality will be determined largely by how well you partition the problem into functions.

Problem

To draw a set of patterns inside a *centered frame* inside a window in accordance with user inputs, detecting and reporting erroneous requests. Let a *pattern* be a 3cm by 3cm grid, composed of nine 1cm by 1cm *blocks*. For this assignment you will work with three pattern types:



Because we will allow the user to specify how much, in terms of quarter (90 degree) turns, the pattern is to be rotated, there are ten possible patterns. (Patt2 has only two distinct 90 degree rotations.) Patterns should be drawn inside an *imaginary, centered frame*, of fixed size 15 cm wide and 12 cm high, inside a window, W, that is 17 cm wide and 14 cm high. By limiting drawing to inside the frame we ensure that no blocks (and therefore patterns) are drawn within 1cm of a window edge. A request to draw a pattern will specify the x and y coordinates for the upper lefthand corner of the pattern (relative to the upper lefthand corner of the *frame*), the pattern to be drawn, and pattern rotation. Rotation should be specified as follows: “U” means no rotation, “L” means 90 degrees counter-clockwise, “R” means 90 degrees clockwise, and “H” means 180 degrees rotation. For example, the prompt and response:

```
Give X,Y coordinates for pattern, pattern type and pattern rotation: 2.75 3.5 1 L
```

requests that pattern one, rotated 90 degrees counter-clockwise, be drawn, with the upper left corner of the pattern 2.75 cm over and 3.5 cm down from the upper left corner of the *frame*.

All pattern positions should be single precision floating point numbers. Portions of any patterns that fall outside of the *frame* should not be drawn, but portions of patterns that fall inside the frame should be. New patterns overwrite portions of the frame they share with previously drawn patterns.

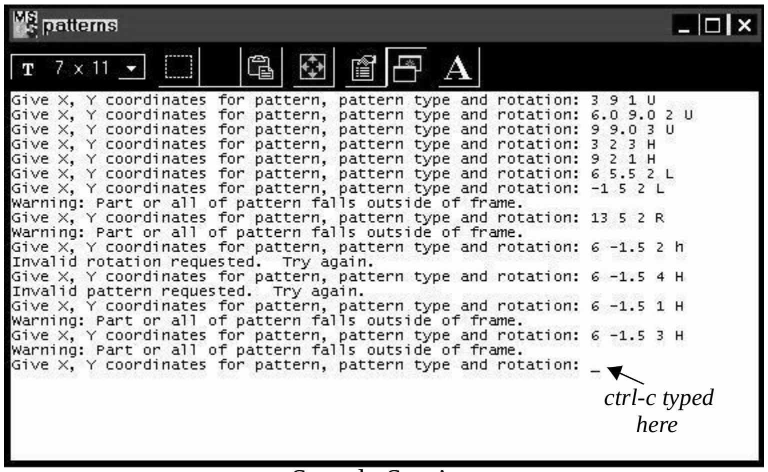
Your program should repeatedly prompt for and read four inputs: the x and y coordinates (single precision, floating point values) of the requested pattern, the type of pattern (integer), and a rotation specification (character). Unless an error prevents it, the pattern should be drawn. Your program should loop until an end of file is encountered (entered by the user) at which time it should terminate. You can cause your program to terminate while within `ApiMain()` by using a return statement—return statements are not limited to being the last statement in a function body.

The errors you should check for and remedies you should apply are:

1. Requested pattern is not 1, 2, or 3: *write error message and request all four pattern inputs again.*

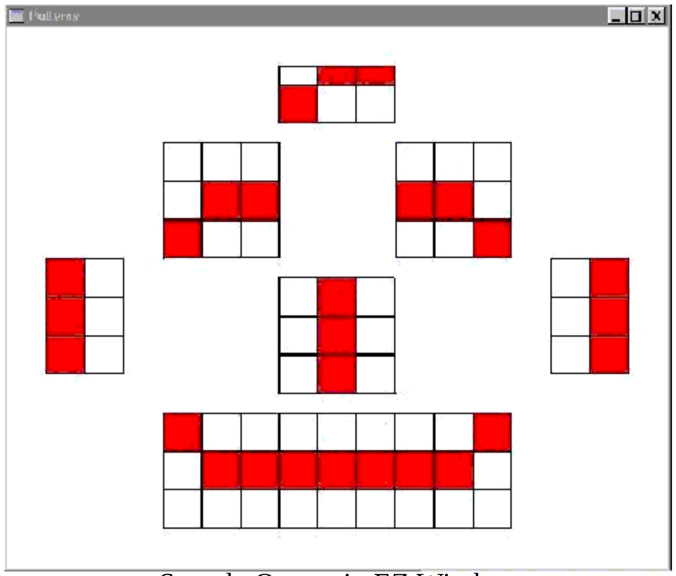
2. Requested rotation for pattern is not “U”, “L”, “R” or “H”: *write error message and request all four pattern inputs again.*

3. Portion (including all) of requested pattern falls outside of frame: *write warning message and draw potion of pattern that falls inside frame.*



```
Give X, Y coordinates for pattern, pattern type and rotation: 3 9 1 U
Give X, Y coordinates for pattern, pattern type and rotation: 6.0 9.0 2 U
Give X, Y coordinates for pattern, pattern type and rotation: 9 9.0 3 U
Give X, Y coordinates for pattern, pattern type and rotation: 3 2 3 H
Give X, Y coordinates for pattern, pattern type and rotation: 9 2 1 H
Give X, Y coordinates for pattern, pattern type and rotation: 6 5.5 2 L
Give X, Y coordinates for pattern, pattern type and rotation: -1 5 2 L
Warning: Part or all of pattern falls outside of frame.
Give X, Y coordinates for pattern, pattern type and rotation: 13 5 2 R
Warning: Part or all of pattern falls outside of frame.
Give X, Y coordinates for pattern, pattern type and rotation: 6 -1.5 2 h
Invalid rotation requested.  Try again.
Give X, Y coordinates for pattern, pattern type and rotation: 6 -1.5 4 H
Invalid pattern requested.  Try again.
Give X, Y coordinates for pattern, pattern type and rotation: 6 -1.5 1 H
Warning: Part or all of pattern falls outside of frame.
Give X, Y coordinates for pattern, pattern type and rotation: 6 -1.5 3 H
Warning: Part or all of pattern falls outside of frame.
Give X, Y coordinates for pattern, pattern type and rotation: _
```

Sample Session



Sample Output in EZ Window

.

Other notes

•Source file `hw04.cpp` is to be completed. Hand in a hard copy of `hw04.cpp` at the start of your regularly scheduled laboratory and submit an electronic version prior to the start of that same lab. No other files should be submitted.

•Your program file should contain the standard header. It should properly identify all members of your group. It must contain the pledge. All group partners must be from the same lab section. There should be only one submission per group.

•When you run the program, it is suggested before typing inputs in reaction to the prompt, that you resize the console window, so that it and the graphical window are both visible and do not overlap.

•The program should follow course programming style guidelines.

•*Think about the design* of this program before doing it. There’s a slightly more clever solution that ultimately requires far less work (and code) than the more obvious solution.