# (Optional) HW 4: Regex with Perl

Due: no later than November 9
(solution posted November 10th)

**Overview:** This is an optional assignment designed to give you some experience with Perl's regular expression capabilities, I/O, and external command execution.

**Objective:** Write a program that

- locates the provided ASCII image pattern in an arbitrarily-sized rectangular ASCII image,
- colorizes the located pattern (and only the pattern) using ANSI escape codes,
- loops through 7 colors forever
- for each color, writes the colorized image to an output file

The ASCII image pattern of interest is:

```
/\_/\
\^x^/
/m m\
```

The pattern above has a width of 5 characters and a height of 3 characters. It may appear at any location in a provided ASCII image. The input ASCII image can be any width or any height, filled with any characters. The given pattern will never be clipped on any sides, and will only appear once in an image.
*Sample input:*

```
o.o/o.o/o.o/o.o/o.o/o.o/o.
o.o/o.o/o.o/o.o/o.o/o.o/o.
o.o/o.a/\_/\o.o/o.o/o.o/o.
o.o/o.a\^x^/o.o/o.o/o.o/o.
o.o/o.a/m m\o.o/o.o/o.o/o.
o.o/o.o/o.o/o.o/o.o/o.o/o.
o.o/o.o/o.o/o.o/o.o/o.o/o.
```

**Guidelines and Restrictions:**

- You *must* read the input image from the `STDIN` filehandle.
- You *must* write the output to a file named `output_colorized.asciimg` (in the same directory) using your own filehandle.

- The output ASCII image *must* be the same dimensions as the input image.
- You cannot use heuristics or looping to locate the pattern of interest; you *must* use regular expressions.

**Helpful hints:**

- Newlines are a single character, but ordinarily won't be matched by `.` in a regexp.
- Consider how in C/C++ a 1D array can represent a 2D array and vice versa. This might help you write your regular expression. (Don't actually try to make a multidimensional array).
- To observe updates to your output file, you can run the `watch` command at a specified interval in another terminal.

  ```
  watch -n.1 -c "cat output_colorized.asciiimg"
  ```

  The `-c` flag retains the output's color, and the `-n.1` runs the command every .1 seconds. You should see the pattern strobe:


- If you read the file output every .1 seconds, you should write to the file roughly every .2 seconds. One way to do this is to `sleep` for this amount of time.
- Your whole script should be roughly 30 lines without comments. Please add comments.
- For reference, the ANSI color codes are:

```
# Regular Colors
Black='\033[0;30m'        # Black
Red='\033[0;31m'          # Red
Green='\033[0;32m'        # Green
Yellow='\033[0;33m'       # Yellow
Blue='\033[0;34m'         # Blue
Purple='\033[0;35m'       # Purple
Cyan='\033[0;36m'         # Cyan
White='\033[0;37m'        # White
```

**Submission**:
Upload a single text file containing the script to Canvas by the due date.