Categories and Subject Descriptors  
K.3.0 [Computers and Education]: General.


Keywords  
Education, assignments, homeworks, examples, repository, library, nifty, pedagogy.

Introduction  
I worry over topics for the syllabus, fretting over demos and presentations. And yet, I always come back to the fact that most of what my students learn and remember from my course comes from the assignments. Great assignments are hard to dream up and time-consuming to develop. With that in mind, the Nifty Assignments session is all about promoting and sharing the ideas and concrete materials of successful assignments.

Each presenter will introduce their assignment, give a quick demo, and describe its niche in the curriculum and its strengths and weaknesses. The presentations (and the descriptions below) merely introduce each assignment. A key part of Nifty Assignments is the mundane but vital role of distributing the materials – handouts, data files, starter code – that make each assignment ready to adopt. The Nifty Assignments home page, http://nifty.stanford.edu, gathers all the assignments and makes them and their support materials freely available.

If you have an assignment that works well and would be of interest to the CSE community, please consider applying to present at Nifty Assignments. See the nifty.stanford.edu home page for more information.

Nick's standing ACM editorial: a large part of the value of Nifty Assignments is that all the material are on the internet for anyone to see without screening people (or search engines!) out by some affiliation or login. In keeping with the ACM's mission to promote the study of Computer Science, the ACM, must find a way to make its materials freely available on the internet. The ACM should raise funds in some other way, such as charging authors.

Copyright is held by the author/owner(s).

SIGCSE '11, March 9–12, 2011, Dallas, Texas, USA.
ACM 978-1-4503-0500-6/11/03.

Image Puzzle (CS0-CS1) - David J Malan, Nick Parlante  
A puzzle based on an image makes a neat form of Media Computation assignment. The following small assignments all require students to write image-manipulation code to solve or reveal something. Depending on the level of scaffolding, the code can be quite short.

Red Puzzle (Malan). Odds are, if your childhood was like ours, you pulled a secret message out of a cereal box at one point that could only be revealed by holding a red sheet of plastic over it. For this puzzle, students are challenged to implement that red sheet of plastic in order to solve a murder mystery! The puzzle allows students to experiment with loops, conditions, and file I/O, and it introduces them to steganography more generally.

BMP Puzzle (Malan). Perhaps the simplest way to represent an image is with a grid of pixels, each of which can be of a different color. For this puzzle, students are challenged to resize BMPs, which store images in precisely that manner. Exposes students to file I/O, hexadecimal, and graphics more generally.

Red Puzzle (Parlante). The Red Puzzle “pattern” also works well in CS0 where the I/O etc. is abstracted and the students just play with the 8 lines or so that manipulates the red channel. The computation can be made quite simple, but the puzzle still has an engaging result and the students get to write the key part of the solution code. This assignment pattern is so nifty and versatile, it should be in every teacher's repertoire.

Blue Puzzle (Parlante). Another CS0 variant on the Red Puzzle, designed to require use of an if-logic. The trick is that the true image is hidden in, say, the blue channel, but only in the values which are in the range 0..16, with all other values being garbage.

Generic Scrolling Game (CS1) - Dave Feinberg  
This assignment proves that a programming project can be very structured without imposing a particular theme/story, allowing students to draw on their creativity to personalize their work. It's perfect for the CS1 teacher who wants to give students an open-ended project, but lacks the manpower to support students facing a variety of coding challenges. Although each student develops a personalized arcade game with their own theme, story, objectives, twists, etc., all students ultimately complete the same methods with essentially similar code, making it easy for the course staff to support and grade.
In the game, the user controls an image on the left edge of the screen, that can be moved up and down. Images are created on the right edge of the screen, and these scroll toward the user's image on the left edge. There are some images that the user tries to get, and there are other images that the user tries to avoid. The user earns a score, and the game ends when some condition is reached. It is up to the students to develop the game's theme/story, by choosing the title, images, distribution of images, game speed, scoring function, game-over condition, and any additional twists.

**Personalized Book Recommendations (CS1)** - Michelle Craig

Virtually every student has had an online experience where a website makes personalized recommendations in hopes of future sales or ongoing traffic. Amazon.com tells you “Customers Who Bought This Item Also Bought”, YouTube makes suggestions for other videos to watch, and NetFlix ran a contest with a million dollar prize for an algorithm that would improve their movie recommendations. In this assignment, students write a program that makes personalized book recommendations using algorithms with increasing levels of sophistication.

After reading a pre-collected set of ratings for a list of books, the program makes recommendations for a particular reader based on a small set of sample ratings from that reader and the preferences of other readers in the community. The assignment was inspired by current machine learning research and provides the opportunity to talk about many other CS ideas including weighted averages, comparing non-trivial objects, designing distance measures between objects and writing comparators, and sparse data.

The assignment is appropriate for the middle of CS1. It uses arrays and reading from files. The solution requires at least a one-dimensional array of strings and either a 2D array of integers or a 1D array of objects.

Students are inspired by the fact that the NetFlix contest offered significant money for the real-life version of this problem. They find it fun to see their programs make personalized predictions for themselves and their friends.

**Wa-Tor World: Predator Prey Simulation (CS1-CS2)** - Mike Scott

A. K. Dewdney described Wa-Tor world in a 1984 Scientific American article. The world is a toroid covered entirely by water. The inhabitants of the world are plankton, fish, and sharks. The fish eat the plankton and the sharks eat the fish. Sharks and fish follow a simple set of rules that produces surprisingly complex behavior. The populations of fish and sharks over time approximate the predicted results of the Lotka-Volterra, or Predator-Prey equations.

I use this assignment in a graphics class for non Computer Science majors. The programming proficiency of the students is similar to students in a CS2 course. I give the students the simulator, based on the APCS GridWorld case study, and they implement a GUI to display the world with fish and sharks, plots of the population of each species over time, and controls to alter various parameters such as how often fish breed and how long can sharks survive without eating. The resulting program output is vibrant and complex. When completed, students and the instructor, tweak the parameters of the simulation and observe the changes.

The assignment could be altered to emphasize different concepts. A CS1 version of the assignment would give students the GUI and the GridWorld code and require them to implement classes for the world, fish, and sharks. A CS2 version of the assignment could require students to create all of the necessary classes necessary for the simulation.

**Hamming Codes (CS2)** - Stuart Hansen

Hamming codes are error-correcting codes. They require $O(lg(n))$ parity bits for $n$ data bits. Each parity bit checks some (but not all) of the data bits. If an error occurs in a data bit, all the parity bits that checked it will show the error, allowing us to uniquely determine where the error occurred and correct it. If two errors occur in the same data block, Hamming codes can detect the errors, but not correct them.

We have used our Hamming code assignment in CS2, but variations can be done in CS1 or a more advanced Data Structures course. Students think the Hamming code assignment is nifty for a variety of reasons. CS Students still like to know how things work at a low level. They find manipulating data at the bit level cool. The assignment integrates matrix multiplication with binary representations of data, two topics typically covered early in a CS curriculum, but often not with an associated programming project. Students also find alternative data representations intriguing. This assignment is in the same spirit as Huffman Trees or LZW compression in that you create an alternative encoding for information. It stretches students' brains into different ways of thinking about encoding data and they like it.

**Evil Hangman (CS2)** - Keith Schwarz

Considering all the effort we computer scientists have spent teaching computers to play games, it's surprising that so little energy has been invested designing software that excels at cheating. Why spend all the effort trying to teach a computer the nuances of strategy when you can simply write a program that plays dirty and wins handily all the time? In this assignment, students build a mischievous program that bends the rules of “Hangman,” repeatedly trouncing its human opponent.

The game Hangman follows simple rules. The computer picks a word, blanks out the letters, and invites the human to guess letters. Whenever the human guesses a letter, the computer then reveals the places in the word where that letter exists. In this “Evil Hangman” variant, the computer does not commit to a single word, but rather maintains a list of every word in the English language of a given length. As the human guesses, the computer parades down this word list, dodging the player's guesses. Once the human has run out of guesses, the computer then picks some arbitrary word from its remaining word list and reveals it as the word it had picked all along. The beauty of this approach is that the human can never prove that the computer was cheating; it just looks like it picked an obscure word and stuck with it the whole way through.

One part of this assignment I especially like is that it has a high ratio of coolness to difficulty. The particular algorithm I suggest to students is straightforward – it can be explained in just a few hours – but it is surprisingly good at trouncing unsuspecting humans. Students often prank their roommates with the program, and I've had several students tell me how they sent the program back to Mom and Dad to show off what they were learning.

This assignment is designed for use early in a CS2 curriculum and gives students a chance to play around with abstract data types such as lists and associative arrays. I have been using Evil Hangman in a CS2-level C++ programming course to help students play around with the STL, though it could easily be adapted for use in Python, C, or Java.