Learning Activities for Secondary and Post-Secondary CS Courses

Peter Andrews, William Slough, Nancy Van Cleave[1]*, Suzanne Westbrook*[2]

Abstract *– In an educational environment meant to serve all students, consideration for diverse learning styles ought to be a prerequisite for our classroom presentations. Effective teachers know it is their concern for each and every student's progress that precedes and makes possible their effectiveness. As most of us are aware, gender and ethnicity data regarding Computer Science degree recipients, as well as the extremely low numbers of minority and female faculty in this discipline, show an atrocious lack of diversity. We feel teaching methods which encourage and support students with diverse learning styles communicates the commitment and concern that faculty have for all students. It is this extra attention, as much as the actual activities themselves, which may make the difference to a student who is becoming discouraged. As an added benefit, these new approaches also make ideas more accessible to students in general since concepts presented with a variety of methods increase each student's chance of understanding.*

Index Terms *– Computer Science Activities, Learning Styles, Minorities in CS, Women in CS*

INTRODUCTION

There have been several reports and much concern regarding the gender and minority inequity in Computer Science [1, 2, 3, 4]. Of course there are a multitude of factors at work in this under-representation. Once students have made it as far as college, these problems may include elements of isolation, low self-esteem, harassment, general lack of sensitivity to relevant issues, lack of role models, and discouragement (both personal and societal) [5, 6, 7, 8]. While a great deal is out of the hands of professors and instructors, there are still some things we can do to encourage women and minorities [9, 10, 11, 12].

The U.S. educational system in this modern age is meant to serve the nation's entire population, but has traditionally given priority and preference to White males [13]. In an educational environment meant to serve all students, consideration of diverse learning styles ought to be a prerequisite for our classroom presentations. Effective teachers know it is their concern for each and every student's progress that precedes and makes possible their effectiveness. We are at our best when we propel our students forward to the point they no longer need us. There is an urgent need to begin addressing the ways we can propel <u>all</u> students forward.

New teaching strategies that encourage and support students with a variety of learning styles also make ideas and concepts more accessible to students in general since they are presented using a range of tools and approaches. The more ways we have of explaining the material, the greater our chances of reaching each student. Fear of failure can inhibit a student's ability to learn, so by making learning less stressful and more enjoyable, we are more likely to retain students who might otherwise drop out or change majors.

TABLE I
DEGREE PRODUCTION BY GENDER

Year	Male	Female	Total
B.S. Degrees in C.S. & C.E.			
93 – 94	6,742 (82%)	1,474 (18%)	8,216
94 – 95	7,561 (82%)	1,372 (18%)	7,561
95 – 96	6,692 (80%)	1,336 (16%)	8,411
96 – 97	6,070 (75%)	1,265 (16%)	8,063
97 – 98	7,773 (77%)	1,568 (15%)	10,161
98 – 99	9,509 (83%)	1,952 (17%)	12,692
M.S. Degrees in C.S. & C.E.			
93 – 94	4,188 (81%)	991 (19%)	5,179
94 – 95	3,554 (80%)	871 (20%)	4,425
95 – 96	3,318 (78%)	852 (20%)	4,260
96 – 97	3,368 (76%)	991 (22%)	4,443
97 – 98	3,748 (76%)	1,094 (22%)	4,934
98 – 99	4,109 (74%)	1,467 (26%)	5,579
Ph.D. Degrees in C.S.			
93 – 94	870 (84%)	140 (16%)	1,010
94 – 95	843 (84%)	163 (16%)	1,006
95 – 96	799 (87%)	107 (12%)	915
96 – 97	763 (85%)	129 (14%)	894
97 – 98	802 (86%)	131 (14%)	933
98 – 99	728 (85%)	124 (15%)	852

CONTEXT

Even a casual survey of our class rosters or our classrooms is generally enough to validate the concern over under-representation in Computer Science. Gender and ethnicity data [14] regarding Computer Science degree recipients and faculty shows a lamentable lack of women and minorities. Table I gives the statistics since 1993-4 on Computer Science degree recipients by gender. The same information is displayed graphically in Figure 1, where it is obvious

[1] Mathematics Department, Eastern Illinois University, Charleston, IL 61920, cfpga@eiu.edu, cfwas@eiu.edu, cfnkv@eiu.edu
[2] Computer Science Department, University of Arizona, Gould-Simpson Bldg., #730, Tucson, AZ 85721, sw@cs.arizona.edu

women fall far below men. The percent of women earning BS and PhD CS degrees remains firmly below 20%, showing very little change in the six years covered by the table. Encouraging, however, is the recent rise in the percentage of Computer Science Master's degrees obtained by women.

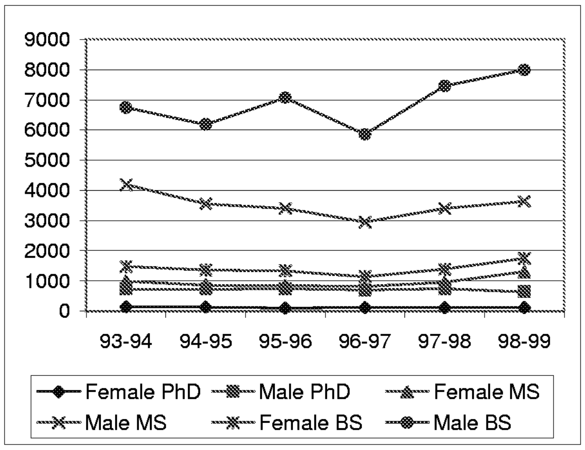

Figure 1.
DEGREE PRODUCTION BY GENDER

Table II tallies the number of faculty at Assistant, Associate, and Full Professor levels by gender. The trends here are equally disturbing since the percentage of females at all levels are low, having made little if any progress, as shown in Figure 2. Without these role models, it will remain difficult to increase the number of women in CS.

TABLE II
GENDER OF PROFESSORS

Year	Male	Female	Total
Assistant			
94 – 95	499 (80%)	125 (20%)	624
95 – 96	434 (81%)	102 (19%)	536
96 – 97	411 (80%)	101 (20%)	514
97 – 98	467 (84%)	92 (16%)	559
98 – 99	624 (84%)	120 (16%)	744
Associate			
94 – 95	872 (90%)	96 (10%)	968
95 – 96	750 (90%)	84 (10%)	834
96 – 97	848 (90%)	100 (10%)	948
97 – 98	861 (88%)	114 (12%)	975
98 – 99	950 (88%)	132 (12%)	1,082
Full			
94 – 95	1,086 (95%)	58 (5%)	1,144
95 – 96	975 (94%)	57 (6%)	1,032
96 – 97	1,044 (94%)	62 (6%)	1,106
97 – 98	1,125 (92%)	92 (8%)	1,217
98 – 99	1,321 (92%)	115 (8%)	1,436

Table III contains minority degree production statistics for the same years as Tables I and II. The numbers are extremely disheartening, although there have been some recent increases.

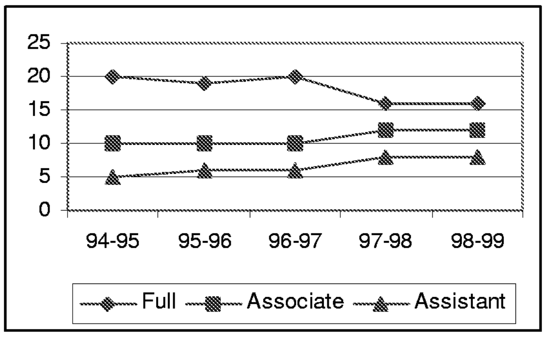

Figure 2.
PERCENT OF FEMALE CS FACULTY

TABLE III
DEGREE PRODUCTION BY ETHNICITY

Year	African American	Hispanic	American Indian	Total
B.S. Degrees in C.S. & C.E.				
93 – 94	172 (3%)	164 (3%)	9 (0.1%)	8,411
94 – 95	152 (3%)	145 (3%)	15 (0.2%)	7,561
95 – 96	207 (2%)	182 (2%)	12 (0.1%)	8,411
96 – 97	157 (2%)	180 (2%)	13 (0.2%)	8,063
97 – 98	251 (2%)	287 (3%)	39 (0.4%)	10,161
98 – 99	327 (4%)	382 (4%)	29 (0.2%)	12,692
M.S. Degrees in C.S. & C.E.				
93 – 94	82 (2%)	64 (2%)	1 (0.02%)	5,189
94 – 95	55 (1%)	52 (1%)	3 (0.07%)	4,425
95 – 96	51 (1%)	39 (1%)	45 (1.1%)	4,260
96 – 97	48 (1%)	66 (2%)	4 (0.09%)	4,443
97 – 98	51 (1%)	41 (1%)	13 (0.3%)	4,934
98 – 99	64 (1%)	50 (1%)	13 (0.2%)	5,579
Ph.D. Degrees in C.S.				
93 – 94	14 (1%)	9 (1%)	0 (0.0%)	1,010
94 – 95	9 (1%)	28 (3%)	1 (0.1%)	1,006
95 – 96	11 (1%)	27 (3%)	5 (0.5%)	915
96 – 97	6 (1%)	8 (1%)	0 (0.0%)	894
97 – 98	10 (1%)	6 (0.6%)	6 (0.6%)	933
98 – 99	17 (2%)	18 (2%)	1 (0.1%)	852

The number of minority CS degree recipients is so low, they must be graphed independently of the other data in order to be visible. Note that in Figure 3, the vertical axis is the total number of students receiving degrees. Although it is promising to see the numbers increasing, they still represent a tiny fraction of the total number of degrees awarded.

The number of minority CS degree recipients is so low, they must be graphed independently from the other data in order to be visible. Note that in Figure 3, the vertical axis is the total number of students receiving degrees. Although it is promising to see the numbers increasing, they still

represent a tiny fraction of the total number of degrees awarded.

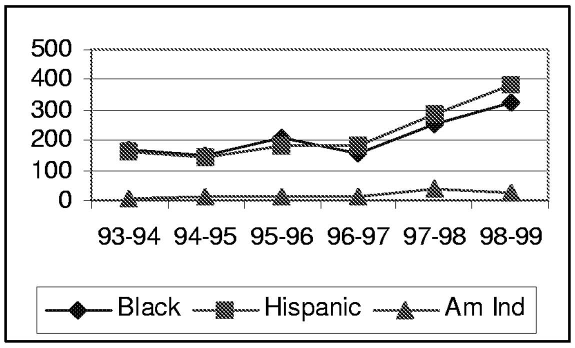

Figure 3.
MINORITY BS DEGREES IN CS

AND THE SURVEY SEZ...

Even a casual survey of our class rosters or our classrooms is enough to validate concern over the low numbers of women and minorities in Computer Science. So how are we to further improve the numbers of women and minorities in CS? To discover what our students thought were characteristics of effective teachers, we did a small survey. We asked several classes to think of the teacher who had had the "most positive impact" on their lives and to write a sentence or two about why they picked that particular teacher. Then we looked for common qualities of these teachers. Overall, there were few surprises in the results. Here are a some representative quotes:

He actually cared if we understood what he was teaching.
He made me think.
He was so excited about teaching he made you want to learn.
The class was as enjoyable as it was beneficial.
Able to teach in so many ways, so the material became clear.
Because she taught the class with much enthusiasm
She took the time to listen to us.
She was really caring and thought so very highly of her students.
He was creative.
Made learning fun!
She really took time to help.
She made learning fun. She gave everyone the sense that if you messed up, that was O.K. and you could fix that problem and learn from your mistakes.
He was real and told us how it really was.

So how do we find ways to show students we really do care about them, that we are wonderful, creative, fun-loving people, and that computer science isn't really so mysterious, when our days (and most nights) are already overbooked? It really doesn't take as much as you might think! Next we present a few possibilities that we hope encourage you to consider ways you can bring a little variety and spice into your classroom, or perhaps add to your repertoire.

THEMES / ACTIVITIES

Although many of the factors related to high drop-out rates and under-represented populations are outside our control, there are still things we can do to make our classrooms more inclusive, less hostile, and places which stimulate intellectual curiosity rather than deadening both the minds and the rears of our students. The following describe some of the interesting activities that our students seem to enjoy, and which often help us evaluate their level of understanding and on what concepts we need to work further.

LANGUAGES AND ENVIRONMENTS

We expect majors in computer science to become competent in several ``industrial-strength'' programming languages before graduating. However, there are a number of other languages and systems such as Logo, ISETL, and Maple which excel in more specialized domains. Although Logo is perhaps best known for its turtle-based graphics, it can be used for a wide variety of general-purpose computing tasks. ISETL (Interactive Set Language) can be used to explore a variety of topics in discrete mathematics. Maple, like Mathematica, Octave, and Derive, is a system for performing a wide variety of symbolic and numeric tasks.

What unites these systems is their interactive nature, which tends to encourage experimentation. We have used Logo to amplify the nature of recursion among beginning students and majors in computer science taking the discrete mathematics course. Recursion with both graphical and non-graphical results can be described in this language with a minimum of attention to syntax details.

Each of ISETL and Maple provide powerful tools which can be used for exploring ideas in discrete mathematics, numerical analysis, and other mathematics and computer science courses. First-hand experiences are often better appreciated than secondhand re-telling common in the lecture format. Consider the problem of estimating a definite integral. Rather than simply telling a class that the rectangle rule is often a poor method, it is better for students to witness this themselves. As a challenge, students are asked to use only the rectangle rule to obtain an estimate to a stated integral, with the aim of obtaining a high-precision estimate. Many students need to witness the time penalty incurred for requesting many thousands of rectangles---and yet get only four digits of accuracy. Given this experience, a follow-up demonstration involving the Romberg method of integration is all the more impressive.

Logo, often thought of as a tool for elementary school education, can be used to display recursion through graphics and color. It is simple to recursively draw a spiral in one color, then change the pen color so students actually observe the unwinding process as the turtle retraces its steps, coloring over the original spiral. This is one way to make a very abstract concept real to students.

ACTING OUT

Problems that students encounter trying to understand sequential programs are magnified when concurrency is introduced. Race conditions and the mechanisms for achieving proper synchronization and mutual exclusion are difficult concepts, even for the best students. Although we do not have a ``magic bullet" for internalizing these difficult topics, we can suggest that class time used for acting out the basic synchronization mechanisms is useful for both students and instructor. Students discover the inadequacies of their understanding—and hopefully fill these gaps—and the instructor gains insight into the level of understanding or misunderstanding among the class.

Recently, one of the authors devoted some class time to acting out the actions taken by a classic solution to the bounded buffer problem. Students were chosen to play the roles of the producer, consumer, and operating system. The buffer was a small area on the front desk; chalkboard erasers were used to symbolize the data. This activity enlivened the discussion of the class and many students realized their understanding of the wait and signal primitives wasn't adequate.

We have also had students in beginning courses execute control constructs (if, pre-test, post-test, and fixed iteration) in small groups. One student decides which path to take or whether the loop continues, while the others represent actions taken in the then and else clauses or in the loop body. To begin, our variables were of type "animal" and we used pictures of animals to represent their contents. This may seem juvenile to some, but overall the students enjoy a break from sitting through lecture or lab, and mistakes in understanding the concepts are easy to see and correct. If you have access to a stairwell, having the students actually step through loops – with the exit test at either the top or bottom of the stairs – sometimes helps students grasp the essence of loops and the difference between the types of loops.

Anther concept which lends itself well to student acting is sorting algorithms such as bubble sort, insertion sort, and selection sort. You can use pieces of string of various lengths, and require students to hold on to both ends (so they can hold only one string at a time). Select a few students to take pieces of strings and turn their backs to the class (so observers do not know the lengths of their strings). Then have one student act as a sorting director to the other students for one of the algorithms they've learned. The director should ask the students to turn around two at a time, while using a third student as a temporary "string holder." (If you have the right students, they'll even go so far as to drop their strings if told to take someone else's before they store their own.)

TOYS AND PROPS

A variety of children's toys can be used to demonstrate many important ideas from computer science. Toys provide various models that can be manipulated by both instructor and students, making abstract ideas more concrete.

Although students don't get the same opportunities for note-taking during these demonstrations, this seems to be compensated by the deeper understanding that they take away from the classroom. Other educators [15] have also reported similar experience to ours.

As a first introduction to recursion, a simple counting problem can be discussed. A small cardboard box, containing some number of objects, is introduced to the class: the objective is to count the number of objects in the box. The primitive operations include testing for zero objects (for example, shake the box), removing a single object, and adding one to a count. For the recursion, we inform students they have *one* friend who can solve a similar problem. Exactly how this friend solves the problem is not important—only that a solution is provided.

Like others, we feel that it is important to distinguish between the recursive design process and the implementation issues. Much later on, if desired, this same discussion could be used to unwind the recursion. In this situation, the friend finds another friend, removing an object and passing the box. For a first exposure to recursion, we feel it is best to delay or avoid altogether the discussion involving the implementation issues. The mental images involved with unwinding the recursion are rarely, if at all, helpful with the thought processes needed for the recursion.

To help students grasp the separate chaining method of hashing, you can use an egg carton cut in half length-wise (so it has a single row of cups) and attach a piece of yarn to the inside of each cup. Using an indelible marker, write numbers on large beads that may be strung on the yarn, then "hash" the beads to the numbered cups and string them on the associated yarn.

Other examples of using props in our classrooms involve using five cards from a deck of playing cards to show insertion and selection sort algorithms. Using interlocking children's blocks (with values written on their sides) to illustrate the recursive merge sort (students "break" a string of blocks in half and pass each half off to be sorted, then merge the sorted blocks). We have labeled small boxes to illustrate inductive proofs, and especially how the *n* in the inductive hypothesis must be arbitrary.

WRITING ACROSS THE CURRICULUM

One technique that is helpful to students who do not necessarily test well is to have them write descriptive stories about concepts such as repetition or parameter passing. Translating these ideas into English involves both left and right brain functions, and students must truly understand what they are to translate before they can write about it. This is also good practice of writing skills for the students. A simple assignment in this vein would be to write three short paragraphs or stories using a squirrel, a pile of nuts, and a storage location the squirrel is attempting to fill with the nuts, which represent pre-test, post-test, and fixed-iteration loops.

Another possibility is to have your students keep journals. Each week you might give them a question they are to answer, or have them describe and give examples for whatever material you're covering in class that week. We have not had the time to implement such idea at this point, and recognize that grading, or even simply reading, a number of these journals could be too time consuming to be feasible.

HUMOROUS AND STUDENT-ORIENTED EXAMPLES

The old tried and true examples are becoming a bit shop-worn in this day and age of global communication and MTV. By the third or fourth time students hear about *queueing* up for theatre tickets, they begin to wonder if computer scientists have any imagination whatsoever, and whether this is the only practical application or purpose for queues!

Even an occasional lousy attempt at humor is better than no attempt at all. It is time to consider a few alternatives to the common analogies we cite in our classroom. For example, we could cite the movie Groundhog Day as an example of (almost infinite) repetition. The action of breaks in a switch statement might bring to mind the song *Viagra in the Waters* by the Four Bitchin' Babes (in a vein similar to *Thirty Thousand Pounds of Bananas*, H. Chapin).

Simple examples may be used, such as shoeboxes for numeric objects (flats for integers, hiking boots for floating point objects) and post office mailboxes for vectors and vectors of vectors. Most students are not so old as to have forgotten about approaching one or the other parent (or both) for money or permission. Good parenting skills (where both parents must assent) can be linked with the Boolean AND, while "not so good parenting," (where only one need say yes), can be linked with the Boolean OR.

MENTORING AND CONCERN FOR STUDENTS

There are many other ways in which we can engage our students and show them respect and encouragement. Food always seems to be a good incentive. One of the authors offers a prize of a candy bar to any student who is first to point out a substantive error or omission in an assignment. The students think it's funny and they do try to find things wrong in order to win the candy bar. It helps make the classroom atmosphere more relaxed and inviting.

Along the same lines, a jar of M&M's can also be used to illustrate recursion. Ask the students how many pieces of candy are in it. Since they'll have to guess, it's passed around with everyone taking one or two pieces until it is empty, then it's handed back through the group with each person adding their total to the previous total until everyone has added theirs in and they have the complete total. Of course they aren't allowed to eat the candy until their count is in.

Have you ever had an afternoon class that seemed to drag on endlessly? The students are tired of sitting through lectures, and as polite as they try to be, some start to nod off in an overheated classroom... One quick and easy solution is to pass around a jar of Tootsie Pops! It is amazing how appreciative students are of such a small gesture, and the sugar buzz helps them stay awake, at least through your class. Candy canes may be substituted during the appropriate time of year.

Several small things that can make big impacts are:

- asking a student how their day's been going, or what their plans are for break or summer, or just noting they look tired
- if you utilize overhead transparencies, use 24 point font, and don't try to pack too much on one foil
- if you do make overheads, shrink them down and print out for the students (any help you can give them with note taking will benefit you as well)
- during the first week of the semester, take digital pictures of your class and put on the internet for local viewing
- encourage students to ask questions (on the first day of class, have everyone raise their hand and repeat "I don't understand," then note that they are still alive and well)
- when a student does ask a question, try to answer it as simply as possible, i.e., using words and examples with which the students are familiar
- encourage students to draw pictures to help them solve problems – especially graphics problems
- don't, however, let yourself get burned out overextending yourself to your students. You must keep a healthy balance in your own life
- remember to smile

CONCLUSION

Some of the keys to improving the classroom climate for students is listening and paying attention to them, communicating our own enthusiasm in the classroom in order to motivate them, making the time to give extra help as often as possible, and holding all our students to high standards. None of the suggestions in this paper are meant to weaken curricula, but to strengthen understanding.

One of the most important things we can teach our students is to ask question. As Dr. Marion Hagler of Texas Tech University once said in a talk to College of Engineering faculty there:

> [W]e try to help students learn three basic things: fundamentals, how to learn, and how to solve problems…
> Learning how to learn includes:
> - learning not to be embarrassed that you do not know everything
> - learning how to decide what you don't know that you need to know
> - learning to ask questions of those who do know, and
> - learning to find and digest new information
>
> If students know the fundamentals and know how to learn, then they can learn whatever they need to know during their career.

Show your students you're concerned for their wellbeing, encourage them to ask questions not just in class but in all aspects of their education, and never stop learning or being excited yourself. The attitude you take into your classroom is often quickly reflected back to you.

REFERENCES

[1] Computing Research Associates, "Taulbee Survey Results," *www.cra.org*, various years.

[2] Camp, T., "The Incredible Shrinking Pipeline," *Communications of the ACM*, Vol. 40, no. 10, pp 103-110, Oct. 1997.

[3] Spertus, E., "Why Are There So Few Female Computer Scientists?," MIT Artificial Intelligence Laboratory Technical Report 1315, *www.ai.mit.edu/people/ellens/Gender/pap/pap.html*, 1991.

[4] Aspray, W., and A. Bernat, "Recruitment and Retention of Underrepresented Minority Graduate Students in Computer Science," Report of a Workshop, *www.cra.org/main/cra.pubs.html*, March 2000.

[5] Bana, S., and Soha Hassoun, "Improving the Graduate School Environment for Women in Computer Science," *A Birds-of-a-Feather Session at the Grace Hopper Celebration of Women in Computing*, *www.cs.washington, edu/homes/soha/GH/list.html*, 1997.

[6] Van Cleave, N., "Components of an American Indian Computer Science Transfer Degree Program," *Journal of Engineering Education*, 1/2001.

[7] Huang, A., A. Ring, S. Toich, and T. Torres, *Gender Relations in Educational Applications of Technology*, Vol. 1, no. 1, 1998, *www-cse.stanford.edu/classes/cs201/Projects/gender-gap-in-education.*

[8] Aspray, "Recruitment and Retention"

[9] ibid

[10] Spertus, "Few Female Computer Scientists"

[11] Van Cleave, American Indian Computer Science"

[12] Margolis, J., A. Fisher, and F. Miller, "Living Among the 'Programming Gods': The Nexus of Confidence and Interest for Undergraduate Women in Computer Science," *www.cs.cmu.edu/~gendergap/confidence.html*, work in progress.

[13] Seymour, E., and Hewitt, N., *Talking About Leaving: Why Undergraduates Leave the Sciences*, Westview Press, Boulder, CO, 1997.

[14] Computer Research Associates, "Taulbee Survey Results"

[15] Bucci, P, Long, T., and Hollingsworth, J., "Toys Are Us: Improving Instruction with Toys in Computer Science Curriculum,"*Workshop, 32nd* SIGSCE Technical Symposium on Computer Science Education, February 21-25, 2001, Charlotte, N. Carolina.