When CS 1 is Biology 1: crossdisciplinary collaboration as CS context

Zachary Dodds
Harvey Mudd College CS Dept.
301 Platt Blvd.
Claremont, CA 91711 USA
00+1+909-607-1813
dodds@cs.hmc.edu

Ran Libeskind-Hadas
Harvey Mudd College CS Dept.
301 Platt Blvd.
Claremont, CA 91711 USA
00+1+909-621-8976
hadas@cs.hmc.edu

Eliot Bush
Harvey Mudd College Biology Dept.
301 Platt Blvd.
Claremont, CA 91711 USA
00+1+909-607-0653
bush@hmc.edu

ABSTRACT
We present the curriculum, deployment, and initial evaluation of a course, BioCS1, designed to serve as CS1 and Biology1 for majors of either (or both) disciplines. Cotaught by professors in both fields, BioCS1 interweaves fundamental biology and computational topics in a manner similar to contextual approaches to CS1. In contrast to other contextual approaches, however, BioCS1 emphasizes both CS and its context equally. The results suggest that cross-disciplinary collaborations can succeed at the introductory level, as they have at later stages of the curriculum.

Categories and Subject Descriptors
K.3.2 [**Computers and Education**]: Computer Science Education

General Terms
Measurement, Design, Human Factors

Keywords
Contextualized CS, CS 1, Biology 1, contextual peers

1. MOTIVATION
Life computes – perhaps no other two-word sound bite better captures the spirit and challenge of modern biology. Life's computation spans orders of magnitude that dwarf those of our artificial machines. Its sophistication and intrinsic value offer a promise to which CS can, at its best, contribute both insight and intellectual resources.

Coordinated approaches to Biology and CS, even in shared spaces such as bioinformatics, now see each field as crucial but independent contributors. Introductory courses remain tethered to parent departments. Believing that tomorrow's Bio and CS fields will be even more interconnected than today's, we have extended work in contextualized CS in order to design, deploy, and assess a novel, shared *BioCS1* curriculum. The key difference from media, robot, and web-based contexts for CS1 [15,19,22,23] is that in our course Biology was not only context but also peer: BioCS1 had to serve as both Bio1 and CS1.

This paper summarizes our BioCS1 curriculum and the student-generated evidence both for and against its effectiveness. In brief, the results show that

- Students gained at least the CS and Biology skills of those in control groups – *for overlapping topics*.
- Students gained algorithm-design and implementation skills *beyond* that of the control group, as motivated by biological context and problems.
- Students taking BioCS1 show an increase in interest, understanding, and excitement in both of those fields.

We necessarily await our jury on several other counts: future enrollment, performance in subsequent CS and Biology courses, and choices of academic major. Here we focus on BioCS1's curricular context, its topics and lab material, and the evaluations of students' knowledge, skills, and affect. We conclude with a vision of how BioCS1 might most usefully – and feasibly – evolve in the future.

2. BIO/CS BACKGROUND
This effort rests on the shoulders of at least a decade-long foundation of collaborative Biology and CS education. Each discipline's futurists foresee deeper links with the other [1,4]; for the moment, bioinformatics dominates the collaboration. Bioinformatics-specific courses [24,25] and programs [5,9,10,11] abound.

In the above-cited curricula, however, the disciplinary merge *follows* students' introductions to the fields. Indeed, this late-curriculum convergence grows appropriately from the specialized subsets of CS and Bio that make up bioinformatics. Other efforts tend to take sides: they offer either biologically-motivated projects for CS students [2,3,7,8] or computational thinking for biology students [16].

A second trend has seen introductory CS courses using biology and genomics as motivating context [12,13]. Such efforts continue to expand and mature [20]. In addition, there are wide-ranging examples of introductory biology curricula that leverage computational tools and increase students' savvy with them [14,26]. Recent examples build with or from biology to span data-analysis skills important for all scientists [6,17]. Importantly, such efforts are not so new that all have succeeded [21]!

ITiCSE'10, June 26–30, 2010, Bikent, Ankara, Turkey.

Perhaps Wellesley's *BiSc303* course [24] is most similar to the course described here: it is also co-taught – in Python – by a biologist and computer scientist with a significant capstone project. Yet *BiSc303* is an advanced elective for majors of both disciplines; to our knowledge, there do not exist other *introductory* courses that interweave biology and computer science topics at a depth and breadth sufficient to fully support majors of either discipline -- either separately or in combination. Indeed, since Bio1 and CS1 are well-established curricula, a single-semester, full-fledged combination might seem impossible.

Yet contextualized CS has shown, compellingly, that adding more to CS1 is possible *with no reduction in students' CS skills and knowledge*. In fact, students' interest and performance can increase appreciably [23]. This work takes the next step: we hypothesize that a *peer*, Biology1, can succeed as a context for CS1. Symmetry dictates that we are making an equally strong statement about Biology1's use of a CS-based context! It is only the CS-themed venue that prompts the first formulation.

But why BioCS1? After all, a contextual peer is a far greater burden. Off-the-shelf contextualized curricula presume that an instructor will quickly pick up the context necessary to motivate the material [22,23]: this does *not* hold for Biology. On the other hand, we believed that the following advantages of interdependence would outweigh our discomfort at losing independence:

1. students' equal or better mastery of CS1 & Bio1 skills
2. students' application of crossdisciplinary thinking
3. students' increased appreciation of both fields
4. students' increased freedom in subsequent courses

The next section summarizes our curriculum; the results then highlight points 1-3, above. Although point 4 will require more time to measure, the final section peeks ahead toward possible futures for Biology/CS collaborations.

3. CURRICULUM AND STUDENT WORK

The setting for this curricular experiment is the BioCS1 class at Harvey Mudd College in fall '09. Twenty-eight first-year students joined this pilot offering, comprising two lectures and a two-hour closed lab per week. This structure is identical to our CS 1 and Bio1 courses, except that Bio1 replaces lab with a recitation section. As we encourage pair-based programming and problem-solving except on exams, we felt that BioCS1's closed labs could adequately serve in lieu of recitation sections, as well.

We took pains to ensure that Biology and CS had equal footing throughout the course. A biologist presented the Monday's lecture each week; the second lecture, by a CS professor, connected a fundamental computational idea with Monday's topic. For the weekly assignments, students applied that computational idea in order solve, model, or explore two to five biological challenges. Python was the computational currency of most of these student homework assignments. We used both a popular biology text [18] and an in-house CS monograph.

Figure 1 presents an overview of our course's material, its integrated presentation as five modules, and a small subset of student homework. Introductions to data and functions offer an opportunity to develop intuition about the alien environment of cell biology: "Do proteins in water feel more like people in a sandbox or people in a ballpit?" The sizes of nucleotide-space (4^n, big) and protein-space (20^n, bigger) are explored computationally, and CS1's traditional assignment exercising conditionals and user-interaction evolved from Rochambeau to *Hydro*-chambeau, in which students wrote programs to reason about molecules' hydrophobicity based on their biochemistry.

topics	Biology	CS	Subset of student HW
wks 1-3	biochemical basics energetics; Gibbs' **ΔG**	data, lists, and functions recursion and functional programming	hydrophilic vs. hydrophobic interactions, implemented as Python/user interactions using `map` to plot rates of enzyme-catalyzed reactions in the presence of different inhibitors
wks 4-6	photosynthesis and cellular respiration metabolic diversity and similarity	combinational circuits memory, registers, and machine language	understanding the Krebs cycle and products designing a one-bit adder in simulation & with physical logic gates; assembly programming
wks 7-9	genomics; proteomics DNA replication and repair mechanisms	the imperative paradigm: loops and accumulators maintaining modularity!	implementing transcription and translation using loops to find coding regions of the genome and introns/exons in lacDNA
wks 10-12	exons, introns, open reading frames from genotype to phenotype: gene-regulation networks	algorithm design: the *use-it-or-lose-it* idiom 2d data, memoization and algorithm efficiency	writing (exponential) knapsack and alignment computing and plotting minimum-energy RNA-folding configurations implementing Conway's Game of Life and extending it to model *Ommatidia* development
wks 13-15	***systems biology***. cellular sensing and signalling via bacterial chemotaxis case study	algorithm complexity and uncomputability	**Final project options**: Build and program a phototaxic robot; fast gene-finding in *Vibrio Cholerae*; modeling HIV as 2d cellular automaton

Figure 1. Summary of BioCS1's syllabus, biological and computational topics, and a small subset of student work [0]

Crucially, *not all topics are integrated across the two disciplines.* They diverge in module 2, where assembly-language programming and circuit-design are presented with photosynthesis and electron transport chains. Yet even there each lecture forges higher-level connections, e.g., modularity and composition among the building blocks from which all circuits, metabolic or synthetic, arise. Though we never had more than one such problem per week, Figure 2 shows HW problem #4 of week 5, a rare example in which no programming is used at all.

1. You have isolated two types of microbe, both of which use hydrogen gas as an electron donor and energy source. One of these uses fumarate as a final electron acceptor, and the other uses nitrate. Which of these would you expect to grow more quickly if you provide them with equal amounts of hydrogen? Explain.

2. Certain fermenting bacteria obtain energy by combining ethanol with water to yield acetate, hydrogen gas and hydrogen ions. This is their main source of energy.

$$2\ \text{Ethanol} + 2\ H_2O \rightarrow 4\ H_2 + 2\ \text{Acetate} + 2\ H+$$

The standard ΔG for this reaction is actually positive (+4.6 kcal/mol). How is it possible for such organisms to obtain energy from a reaction with a positive ΔG?

3. Feedback Inhibition. Citrate is the first intermediate in the citric acid cycle. When the citric acid cycle is slowed, the concentration of citrate builds up in the cell.

How would you expect the build up of citrate to affect the activity of the following enzymes? Explain the logic behind each answer.

- Phosphofructokinase, an allosteric enzyme that catalyzes the conversion of fructose-6-phosphate to fructose-1,6-biphosphate, an early step in glycolysis?
- Acetyl-CoA carboxylase an allosteric enzyme which catalyzes the first step of the metabolic pathway going from Acetyl-CoA to fatty acids?

Figure 2. This screenshot shows problem 4 of homework 5 in BioCS1. Not all coursework uses both Biology and CS.

The focus shifts from monomer to polymer in module 3, with imperative programming growing out of module 2's low-level computation. Loop-and-counter idioms, in turn, allow students to implement transcription and translation, Bio and CS's most ubiquitous shared topics. More nuanced biological understanding then motivates more sophisticated computational applications:

tracking open reading frames (ORFs) despite introns and exons and determining gene presence via the expected lengths of ORFs.

Module 4 departs from our traditional CS1 curriculum to exploit the opportunity to build skills and intuition in algorithm design. We emphasize the *use-it-or-lose-it* strategy that recursively - and exponentially - compares solutions in the case that the input's first element ***is*** used and the case in which it ***is not*** used. As CS education terminology is not standardized here, Figure 3 provides an example of student-written code from week 8's closed lab.

```
def subset1(P,L):
  """Returns true if it is possible to
     make number P as the sum of elements from
     list L. Input integer P and list L"""
  B = sorted(L)
  C = B[::-1] #puts the list in DESCENDING order
  if P == 0: return True
  elif C == []: return False
  elif C[0]>P:
    return subset1(P,C[1:]) # Splices it out if it's too big
  else:
    LoseIt = subset1(P,C[1:]) #You've lost it
    UseIt = subset1(P - C[0],C) #Keep C intact here
    return UseIt or LoseIt
```

Figure 3. An example of use-it-or-lose-it problem solving by a student with no CS/programming background before BioCS1. This Python code here is unaltered from week 8's submitted version except that its spacing has been slightly compressed.

Students also write such exponential solutions for global sequence alignment and minimum-energy RNA folding (the latter with a graphical interface using Python's **turtle** package). Because the course is also Biology1, they leverage their programs to draw conclusions about phylogenetic relationships. Also, the frustrating slowness of their solutions does more than any CS lecture to motivate the speedups available through *memoizing* calls. That approach, in turn, prompts 2d data structures, students' implementation of Conway's *Game of Life*, and its extension to modeling ommatidia (eye-facet) development in *Drosophila* through lateral inhibition and state-change.

The final module runs concurrent with three medium-sized capstone projects from which the students choose. An HIV-modeling option stretches students' data-structure familiarity and extends 2d cellular automata; a gene-finding project builds module 4's small exercises into full-organism analysis and classification: speed is of the essence! A robot-building and programming option ties module 2's low-level computation into module 5's systems biology emphasis. As lectures present the sensing, signaling, and methylation-based adaptation of flagellar chemotaxis, students implement analogous phototaxic behaviors on a hand-built robot. Module 5's computational lectures segue from the efficiencies of memoization to complexity and several examples of uncomputability.

Topic Tradeoffs What did BioCS1 *lose* relative to Bio1 and CS1? Its students did miss a great deal: compression, image-manipulation, writing the DFT to analyze sounds, Markov-text generation, the Mandelbrot set, a web-based TextClouds project, and a Connect-4 tourney, to name a few from CS1. Population biology, some innovations in laboratory techniques, and some context of biological breakthroughs were also postponed. Yet rather than focus on such "losses," we feel BioCS1's curriculum is a win for both Bio and CS because it *adds* to the corpus of examples with which each field can engage its students.

3.1 Laboratory Sessions

The course offers students a regularly-scheduled closed lab. Most weeks, lab is a low-pressure setting in which to review biological concepts while gaining confidence with the Python required to investigate the week's homework. Because our Biology1 course does not offer wetlabs, neither does BioCS1. Yet we did intersperse many "drylab" activities with which we sought hands-on *computational* metaphors to reinforce BioCS1's primary theme: *life computes*. Figure 4 highlights these.

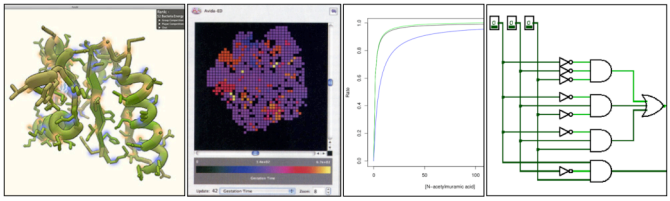


Figure 4. Nonprogramming lab activities in BioCS1. (L-R): FoldIt, Avida-Ed, enzyme kinetics plots, Logisim. Below, we describe our Arduino-based physical circuit construction.

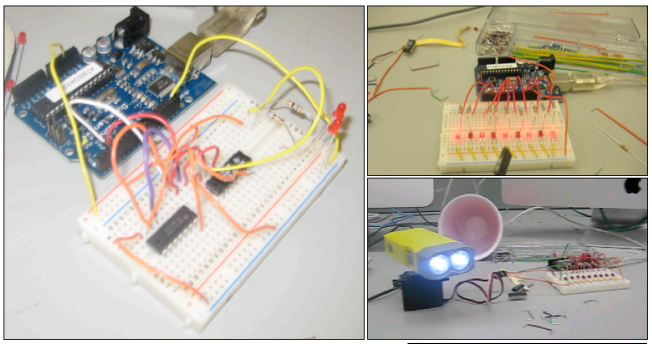
Within this sequence of closed labs, we identified two opportunities for which hands-on, physically embodied computation seemed particularly appropriate:

- To reinforce module 2's ideas of modularity and composition in biological and artificial computation, we wanted students to build simple physical circuits from logic gates -- and then put those circuits to use in a tabletop light-seeking task.
- When presenting the chemical basis for single-celled organisms' volition in module 5, we hoped students could *physically* model the directed random walks produced by changes in flagellar rotation.

Certainly we could have provided an off-the-shelf circuit-building kit for the former lab and a prebuilt robot to support the latter final project option. Yet distinct hardware platforms each require their own, often significant, learning curve. More importantly, using a separate tool for each project unnaturally hides an insight common to real biological and real computational systems: the layers of abstraction that make such complex systems possible. Although we might argue their relative sophistication, the hierarchy that creates ecosystems from elements and the one that creates Google from gates share all the challenges of modularity, interdependence, and staggering depth.

Thus, we opted to explore *single* electronic tangibles that might scale through the term. Seeking simplicity, ease of Python programmability, and low cost, we settled on Figure 5's Arduino-based materials for these two physical-computation labs. As a result of this decision, the 13 students opting to implement a phototaxic model of bacterial behavior did so via their own hand-built robot platforms. Step-by-step guidance on these materials and running the labs appears with the complete course at [0].

These BioCS7 hardware resources had an immediate impact on our simultaneously-running introductory CS course: students in CS1 also wanted to gain experience with physical circuit construction and control. To meet that demand, we hastily scheduled five completely optional two-hour labs. Because these sessions attracted 73 attendees from CS1's 190 students, these hardware labs will become an official part of CS1 in the future.



Materials List
for 2 students together

Arduino	$30
USB cable	$ 5
Protoboard	$ 5
Servo motor	$12
Logic Gates	$ 1
Photoresistor	$ 1
Wire and LEDs	$ 3
total	$57

Additional Materials
for the chemotaxis-modeling robot

+ servo motor	$12
+ photoresistor	$ 1
and when untethered	
+ batteries	$ 5
+ battery holder	$ 2
total	$20

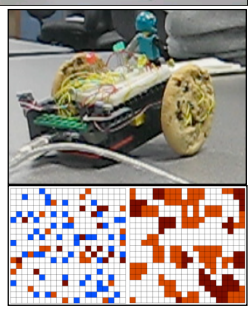


Figure 5. [top] Light-lab subprojects showing the components used, including circuit composition in the one-bit adder at left, LED outputs in students' "hypnotizers," top right, sensing via a photoresistor, and actuation through a continuous-rotation servomotor (lower right). [bottom] Parts and prices of those labs' materials, along with *Cookiebot*, one of 7 final-project robots, and two generations of a student's HIV-modeling CA.

4. EVALUATION AND RESULTS

Ultimately, it is student development that determines the success or failure of crossdisciplinary efforts such as this. That is, can students in BioCS1 exhibit both CS skills and Biology skills to the extent that peers in CS1 & Bio1 can?

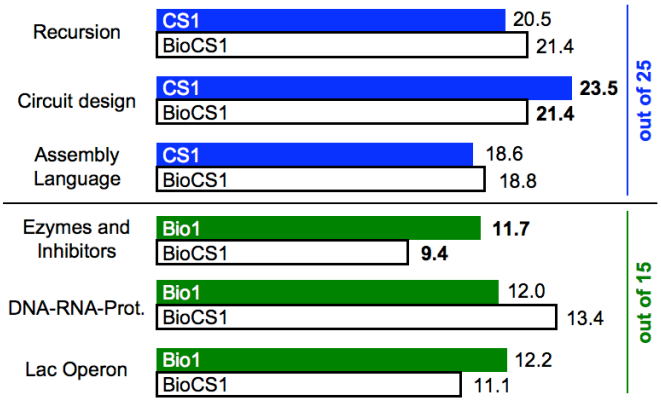


Figure 6. Comparison of identical CS exam questions and near-identical Biology questions. Highlighted in bold are the differences significant at the $t < .05$ level: circuit design and enzymes. In both of these cases we note that BioCS's much richer experience did *not* appear in the assessment.

Shared CS and Bio work We placed three identical computational questions on the CS1 and BioCS1 final exams. We also compared nearly-identical questions between Bio1 homework problems and BioCS1 hw/exam questions. Figure 6 summarizes the performances of each cohort. Only two differences significant at the $t < .05$ level emerged: circuit-building and enzyme/inhibitor analysis. In the latter case, Bio1 students submitted a take-home assignment, but the BioCS1 students answered a question on a timed exam – the delivery may have contributed to the differences. Although future coursework will elucidate these data, Figure 6's similarities especially encourage us because those CS questions were far *less* biological than BioCS1's coursework and because BioCS1's students were all first-years, but Bio1 had only sophomores or beyond.

Additional skills In the same breath as Figure 6's comparisons, we should point out that BioCS1 students developed and exercised a set of skills *above and beyond* CS1 or Bio1 students. In fact, the curriculum of BioCS1 comprises only about 80% of CS1 and 80% of Bio1: the module on algorithmic development comes in lieu of CS1's equal-sized module introducing OO programming. The system-biology module similarly replaces three weeks of population biology in Biology1. The most meaningful assessment of such incommensurate differences will come from our tracking of student work in future courses.

Affective outcomes Particularly at our school, where students do not choose a major until their second year, course choice offers an important barometer of student affect. That 21 of the 28 students in BioCS1 chose to take the now-underway BioCS2 course offers a strong, if not unanimous, vote of confidence in the BioCS1 experience. In fact, interviews revealed that 6 of the 7 students headed elsewhere switched not from dissatisfaction with BioCS1, but from rapidly developing interests in other disciplines. Anonymous feedback reinforced this message, including *"labs are awesome"* and *"I like how bio and cs are so closely linked."* First-offering jitters also showed, however: *"sometimes the integration of bio and cs seems a bit forced or unrealistic"* and *"rethink and unify terminology."*

Workload We sought to layer Biology and CS in the low-overhead spirit of many contextualized CS offerings, but we only *almost* succeeded in keeping workload consistent. Figure 7 summarizes anonymous surveys from CS1 and BioCS1 students, all of whom share courseloads. (The sophomore schedules of Bio1 students are different.)

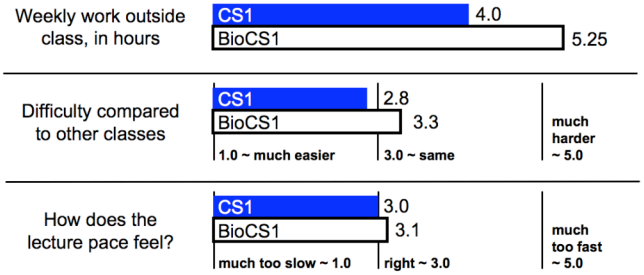


Figure 7. Student-reported workload data from CS1 and BioCS1 had differences significant at the $t < .05$ level. Lecture pace and perceived difficulty, slightly higher, did not differ significantly from those of the control group, all first-years.

Confounding factors Placement is one confounding factor we have wrestled with: all BioCS1 students opted to take the course, but only *most* of the CS1 students did: 22 of the 50 interested in BioCS1 were placed into CS1 for lack of space. In both CS1 and in BioCS1 a subset of students arrived with some computational background and a larger subset had none (whereas almost all of both groups did have high-school biology). Yet we do not know the *relative* sizes of those subsets.

Conclusions We feel that even the most conservative conclusion we might draw – that students must *choose* BioCS1 to succeed in it – opens exciting and largely unexplored opportunities for the computational education of coming generations of scientists.

5. PERSPECTIVE

As with any initial offering, this BioCS1 course did have its rough edges! We believe those transient effects do not detract from the course's primary message: *that full-fledged crossdisciplinary collaborations can succeed as CS1 contexts*. Conversely, the evidence suggests that CS can also succeed as a context for introductory biology. We plan to offer a smoother-edged BioCS1 with twice as many students in Fall 2010. In addition, we are tracking 2009's students through CS, Biology, and other academic choices they make in the next three years.

Curricular combinations such as BioCS1 help confirm that, far from detracting from or "displacing" material, computation can *enhance* the knowledge, contributions, and disciplinary-specific thinking that Biology1 seeks to convey. We hope this effort sparks similar peer-as-context approaches in chemistry, engineering, mathematics, physics, and beyond. We look forward to an era of integrative science education in which computation can act as both an effective collaborator and an inspiring catalyst.

6. ACKNOWLEDGMENTS

The authors thank HHMI award #52006301, NSF DUE CCLI #0536173, and Harvey Mudd College for their generous support.

7. REFERENCES

[0] Course URL: www.cs.hmc.edu/twiki/bin/view/CS6/Fall2009HW

[1] 2020 Science Group. 2005. Towards 2020 Science Microsoft.

[2] Adams, J, Matheson, S, and Pruim, R. 2008. Blasted: integrating biology and computation. *J. Comp. Small Coll.* 24(1): 47-54.

[3] Beck, J, Buckner, B, and Nikolova, O. 2007. Using interdisciplinary bioinformatics undergraduate research to recruit and retain CS students. *Proc. SIGCSE 38* ACM Press: 358-361.

[4] Board on Life Sciences. 2003. Bio 2010 Nat. Acad. Press.

[5] Bruhn, R and Jennings, S. 2007. A multidisciplinary bioinformatics minor. *Proc. SIGCSE 38* ACM Press: 348-352.

[6] Burhans, D and Skuse, G. 2004. The role of CS in undergraduate bioinformatics education, *Proc. SIGCSE 35* ACM Press: 417-421.

[7] Cutter, P. 2007. Having a BLAST: a bioinformatics project in CS2. *Proc. SIGCSE 38* ACM Press: 353-357.

[8] D'Antonio, L. YEAR. Incorporating Bioinformatics in an algorithms course *SIGCSE Bulletin*, ACM Press 35(3): 211-214.

[9] Doom, T., Raymer, M., Krane, D., and Garcia, O. 2002. A proposed undergraduate bioinformatics curriculum for computer scientists. *Proc. SIGCSE 33* ACM Press: 78-81.

[10] Goode, E. and Trajkovski, G. 2007. Developing a truly interdisciplinary bioinformatics track: work in progress. *J. Comput. Small Coll.* 22(6): 73-79.

[11] Khuri, S. 2008. A bioinformatics track in computer science, *Proc. SIGCSE 39* ACM Press 508-512.

[12] LeBlanc, M. D. and Dyer, B D. 2004. Bioinformatics and CC2001: why computer science is well positioned in a post-genomic world, *SIGCSE Bull.*, ACM Press 36(4): PAGES.

[13] LeBlanc, M. D. and Dyer, B D. 2003. Teaching together: A three-year case study in genomics. *J. of Comp. Small Coll.* CCSC Press 18(5): 85-95.

[14] McGuffee, J. 2007. Programming languages and the biological sciences, *J. of Comput. Small Coll.* 22(4): 178-183.

[15] Pearce, J. and Nakazawa, M. 2008. The funnel that grew our CIS major in the CS desert. *Proc. SIGCSE 39* 503-507.

[16] Qin, H. 2009. Teaching computational thinking through bioinformatics to biology students. *Proc. SIGCSE 40* 188-191.

[17] Robbins, K. 2010. vip.cs.utsa.edu/classes/cs1173f2009

[18] Sadava, D., Heller, H, Orians, G, Purves, W, and Hillis, D. Life: The Science of Biology, W. H. Freeman and Co. NY, NY.

[19] Schaub, S. 2009. Teaching CS1 with web applications and test-driven development. *SIGCSE Bull.* 41(2): 113-117.

[20] Soh, L-K, et al. 2009. Renaissance computing: an initiative for promoting student participation in comp., *SIGCSE 40* 59-63.

[21] Stone, J. A., Medica, D. L., and Fetsko, L. A. 2009. Experiences with a CS1 for the health sciences. *SIGCSE Bull.* 41(2): 122-126.

[22] Summet, J., Kumar, D., O'Hara, K., Walker, D., Ni, L., Blank, D., and Balch, T. 2009. Personalizing CS1 with robots. *SIGCSE Bull.* ACM Press 41(1): 433-437.

[23] Tew, A E, McCracken, W M, and Guzdial, M. 2005. Impact of alternative introductory courses on programming concept understanding. *Proc. ICER 1* ACM Press: 25-35.

[24] Tjaden, B. 2007. A multidisciplinary course in comp. biology. *J. Comput. Small Coll.* CCSC Press 22(6): 80-87.

[25] Toth, C. and Connelly, R. 2006. A bioinformatics experience course. *J. Comput. Small Coll.* CCSC Press 21(6): 100-107.

[26] Wray, K.A. 2005. Perl algorithm to calculate and categorize φ and ψ angles in a protein. *J. Comp. Sci. in Coll.* 20(5): 98-99.