Bio1 as CS1: Evaluating a Crossdisciplinary 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 an introductory course in *both* biology and CS. Co-taught 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 such cross-disciplinary collaborations can thrive at the introductory level, just as they have later in 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 aphorism better captures the spirit and challenge of modern biology. Life's computation spans orders of magnitude that dwarf that 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.

Combined approaches to Biology and CS, even in shared spaces such as bioinformatics, treat each field as crucial but independent contributors. Specifically, introductory courses remain tethered to parent departments. Believing that Biology and CS will be even more interconnected in the future, we have run a three-year experiment with a novel, shared *BioCS1* curriculum. Distinct from media, robot, and web-based contexts for CS1 [16,20,23,24], in BioCS1 Biology was not only context but ***peer***: the course serves as both Bio1 and as CS1 for majors of either field as well as for non-majors.

ITiCSE'12, July 3–5, 2012, Haifa, Israel.

This paper reviews BioCS1's curriculum. We then present the student-generated evidence of its effectiveness. In brief, the results show that:

- Students in BioCS1 gained the same level of CS skills as a natural control group – and the entire cohort – of their peers in the standard CS 1 course.
- BioCS1 stimulated equal or greater student interest in both computing and biology as measured by (1) anonymous course evaluations, (2) the decision to take CS2, and (3) the choice of a biology or CS major course of study.

We also report on the student workload and affective response to BioCS1. 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 now, bioinformatics dominates the collaboration. Bioinformatics-specific courses [25,26] and programs [5,9,10,11] abound.

In the above-cited curricula, however, the disciplinary merge *follows* students' introductions to the fields. This late-curriculum convergence stems naturally from the specialized subsets of CS and Biology that make up bioinformatics. Other efforts tend to be asymmetric: they offer biologically-motivated projects for CS students [2,3,7,8] or computational thinking for biology students [17].

A second trend has seen introductory CS courses using biology and genomics as motivating context [12,13]. Such efforts continue to expand and mature [21]. In addition, there are wide-ranging examples of introductory Biology increasing students' computational savvy with domain-specific tools [14,27]. Recent examples also build with or from biology to span data-analysis skills important for all scientists [6,18]. Importantly, such efforts are not so new that all have succeeded [22]!

Perhaps Wellesley College's *BiSc303* course [25] is most similar to the course described here: like ours, it is co-taught – in Python – by a biologist and computer scientist and has 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 CS at a depth and breadth sufficient to fully support majors of either discipline. Indeed, since Bio1 and CS1 are solidly 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 [24]. This work takes the next step: we hypothesize that a *peer*, Bio1, can succeed as a context for CS1. Symmetry dictates that we are making an equally strong statement about Bio1'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 [23,24]: this does *not* hold for Biology. Yet we believe that the following advantages of interdependence far outweigh our discomfort at losing independence:

1. students' equal or better mastery of CS1 & Bio1 skills
2. students' application of cross-disciplinary thinking
3. students' increased involvement in both fields
4. students' time-efficiency: two courses in one class

The next section reviews BioCS1's curriculum, emphasizing differences from the pilot three years ago [0]; the results then highlight points 1-4, above. We conclude with several lessons that this BioCS1 experiment has taught us.

3. CURRICULUM AND STUDENT WORK

The setting for this curricular experiment is a BioCS1 class that has now been taught three times -- in fall terms of 2009, 2010, and 2011 -- to a total of approximately 100 students. BioCS1 is a 14-week semester course comprising two 75-minute lectures and a two-hour computing lab per week, a structure similar to our CS1 and Bio1 courses.

All of our undergraduates, regardless of intended major, must take an introductory CS course in their first semester at the college. Students with prior experience can place into accelerated or advanced courses, but the majority take CS1. BioCS1 is a relatively recent addition to the curriculum and can be taken in lieu of CS1. Students indicate their course preferences and are placed accordingly. However, BioCS1 is intended exclusively for students with *no prior computing experience.*

By design, Biology and CS have equal footing in BioCS1. A biologist presents the first lecture each week and a computer scientist presents the second. The two lectures are tightly integrated in support of that week's topics. Students apply the week's computational idea in order to solve, model, and explore two to five biological questions for each week's homework assignment. As in CS1, BioCS1 students use Python throughout.

Module 1 Figure 1 presents an overview of BioCS1's topics and a small subset of student homework. The course is made up of seven two-week modules. By the end of the first module, students have written their own programs to simulate a bacterium's random walk in a Petri dish using the `turtle` package (a topic revisited with increasing sophistication through the term) and to implement the Central Dogma of Biology: the transcription and translation of DNA to proteins.

Module 2 increases comfort with computational problem-solving, asking students to write their own programs to simulate important biological ideas like genetic drift and natural selection. These fundamental processes become more concrete and meaningful when students write their own programs to model them and explore the effects of their parameters.

	Biology	CS	Subset of student HW
weeks 1-3	DNA, RNA, genes, and the central dogma	introduction to Python	gene-finding and modeling gene expression
weeks 4-5	population genetics + molecular evolution	randomness and Monte Carlo simulation	population genetics simulations
weeks 6-7	sequence alignment	recursion and larger-program design	sequence alignment algorithms
weeks 8-9	phylogenetics	recursion on trees	estimating phylogenetic trees from sequence data
weeks 10-11	structure and function: RNA folding	memoization	implementing polynomial-time RNA folding
weeks 11-12	systems biology: modeling chemotaxis	software modeling via objects and classes	modeling chemotaxis using Python's turtle graphics libraries
weeks 13-14	neuroscience	uncomputability	student capstone projects

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

Module 3 provides a first exposure to recursion in the context of sequence alignment and other related problems. Questions like *Are humans more closely related to chimpanzees or to gorillas?* pique students' interest and motivate algorithms for determining the differences between genetic sequences. In this module, students design and implement a set of increasingly sophisticated recursive functions and apply them to small biological datasets.

Once students can compute the differences between different species' genes, these differences can be used to infer putative evolutionary or *phylogenetic* trees. **Module 4** explores both the biological basis of these phylogenies and the recursive algorithms to estimate them. Figure 2, left, shows one student-generated example. Here, the recursion is on trees – the data structures, not arboreal ones! – a topic that is also taught in CS1 course, but without the rich motivating application that biology offers.

Although recursion is an effective tool to explore data and answer biological questions, it is not an efficient one: the algorithms students develop have exponential running time. **Module 5** explores the efficiency of algorithms, including the difference between exponential and polynomial time, and a glimpse at NP-completeness.

Students then learn *memoization*, a form of dynamic programming that reinforces dictionary data structures and allows many of their exponential algorithms to run in polynomial time. As a result, students can handle data sets of a size relevant in current biological work. Memoization also allows investigation of biologically fundamental questions, e.g, *How does RNA fold?*, as depicted graphically and programmatically in Figures 2 and 3. This coupling of powerful CS techniques with compelling biological applications symbiotically motivates both disciplines in a way that would be difficult – perhaps impossible – to achieve in a CS1 or Bio1 course alone.

Module 6 introduces Systems Biology, whose mathematical models and computational implementations yield insights into complex biological systems. BioCS1 students build a classical

chemotaxic model that explains how a bacterium can use chemical gradients to move toward a source of food. The interacting agents of systems biology are best modeled through object-oriented programming, and students build their own classes and objects to do so.

Module 7 concludes the semester with several fundamental CS questions. First, we examine basic ideas in neuroscience and what it means for biological systems to compute. Concurrently, we explore elementary ideas in the *limits* of computability, ultimately establishing the existence of uncomputable functions. During this module, students choose one of three medium-sized capstone projects for the course. Each option exercises the ideas of modularity, classes, and objects introduced in the previous module: one option returns to cell biology, modeling interactions and catalysis within an artificial chemistry; the second uses likelihood methods to deduce the structure of gene regulatory networks from micro-array data (Figure 2, at right); the third option uses genetic programming to optimize the behavior of a Karel-like automaton [15].

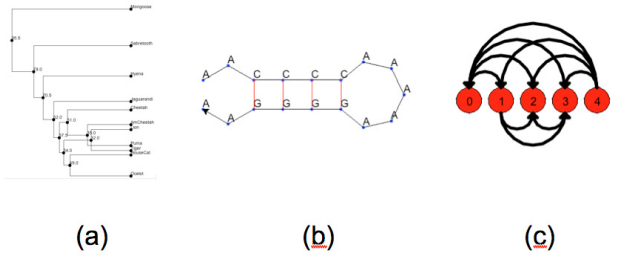

Figure 2. Example images from student work in BioCS1: (a) a deduced phylogenetic tree (week 9), (b) a computed fold within an RNA molecule (week 11), and (c) a maximum-likelihood gene regulatory network (week 14). Each image was created by a student submission that computed the structure in question and rendered via Python's turtle-graphics library.

Does BioCS1 cover all of CS 1 and Bio1? BioCS1 is a one semester course. In 2009-2010, we offered a follow-on "BioCS2" course. Of the 28 students who enrolled in BioCS1, 21 chose to take BioCS2. Students who took both of these courses saw the foundational topics in our one semester CS 1 course and our one semester Bio 1 course and satisfied both the college's CS and biology requirements.

In academic years 2010-11 and 2011-12, we offered BioCS1 without the BioCS2 follow-on course. In one semester it is clearly not possible to offer all of the material in CS 1 *and* Bio1. However, CS1 material not covered in BioCS1 is largely contextual, not fundamental. The CS1 course exposes students to a breadth of application areas, e.g., games, AI, and cryptography, in contrast to the single context that BioCS1 offers.

On the biology side, BioCS1 provides a first broad exposure to college-level biology by surveying some of the major ideas in the field and offering students weekly "dry lab" experiences that are not offered in the school's large Bio1 course. On the other hand, Bio1 does cover topics in both molecular and population biology absent from BioCS1. Even so, many students who complete BioCS1 can review those topics and successfully place out of Bio1. In 2010-11, every BioCS1 student placed out of Bio1, as did the vast majority of those who sought to pass out in 2011-12.

```
def compBase(base):
    """Returns the complementary RNA base."""
    if base == 'A': return 'U'
    if base == 'U': return 'A'
    if base == 'G': return 'C'
    if base == 'C': return 'G'

def adjust(pairs, k):
    """Takes a tuple of number pairs and
    increases the numbers by k."""
    output = ()
    for pair in pairs:
        adjPair = ((pair[0]+k, pair[1]+k),)
        output = output + adjPair
    return output

D = {}

def getFold(RNA):
    """ Returns an optimal folding of the given
    RNA string."""
    if len(RNA) <= 1: return (0, ())
    elif D.has_key(RNA): return D[RNA]
    else:
        maxUseIt = (0, ())
        for base in range(1,len(RNA)):
            if RNA[base] == compBase(RNA[0]):
                Int = getFold(RNA[1:base])
                IntPairs = adjust(Int[1], 1)
                Ext = getFold(RNA[base+1:])
                ExtPairs = adjust(Ext[1], base+1)
                useIt = (1 + Int[0] + Ext[0],
                         ((0, base),)+IntPairs+ExtPairs)
                if useIt[0] > maxUseIt[0]:
                    maxUseIt = useIt
        loseIt = getFold(RNA[1:])
        loseItPairs = adjust(loseIt[1], 1)
        if maxUseIt[0] >= loseIt[0]:
            RNAmax = maxUseIt
        else:
            RNAmax = (loseIt[0], loseItPairs)
        D[RNA] = RNAmax
        return RNAmax
```

Figure 3. A student's solution to the RNA folding problem of week 11, using recursion with memorization for efficiency.

4. EVALUATION AND RESULTS

Ultimately, it is student development that determines the success or failure of cross-disciplinary efforts such as this. In order to assess the choices and capabilities of the students taking BioCS1, we have tracked the first two cohorts (those taking BioCS1 in 2009 and 2010) through several measures of computational proficiency, through reported workload and affective feedback, and through subsequent choices of major. Because students at our institution do not choose a major program of study until the middle of the second year, we do not include the most recent cohort (2011) in these data. We are continuing the evaluation of BioCS1, however, and we do include that latest group where applicable, below.

Small-scale comparisons

In Fall 2009, we placed three identical fundamental computational questions on the CS1 and BioCS1 final exams. We also compared nearly identical questions between Bio1 homework problems and BioCS1 homework/exam questions. Although the details are reported elsewhere [0], the important outcome of these head-to-head problems was that student performance in BioCS1 did not differ significantly from that of students in Bio1 or students in CS1.

In the most recent cohort, Fall 2011, BioCS1 and CS1 shared one of the capstone projects (a genetic programming project). This capstone project requires a synthesis of a number of foundational CS topics including basic data structures and data types (e.g., two-dimensional arrays, dictionaries), object-oriented design, and,

most importantly, the design, implementation, and testing of a mid-sized program (approximately 200 lines of code) from scratch. The BioCS1 and CS1 students performed equally well on this project, suggesting that both groups attained the desired level of computing proficiency.

	BioCS1 (2009-11)	CS1 Control (2009-11)	All students (2009-11)
cohort size	63	38	384
fraction taking CS2	49.2% (31/63)	44.7% (17/38)	44.0% (169/384)
average CS2 grade	3.69 (σ = 0.37)	3.53 (σ = 0.40)	3.56 (σ = 0.55)
choosing CS major	19.0% (12/63)	18.4% (7/38)	21.4% (82/384)
choosing Bio major	27.0% (17/63)	7.9% (3/38)	7.0% (27/384)

Figure 4. Data on how BioCS1 students fare in CS2 relative to the control cohort (those who wanted to join BioCS1 but could not because of space limitations) and all of the students in the 2009-2011 classes. In addition, we compare the rate at which BioCS1 students choose CS or biology majors with the control and institution-wide statistics. Note that BioCS1 students continue to take more CS and choose both CS and biology as a major more often than the control, though not significantly so. In addition, their performance in CS2 is *better* than both the control group's and the whole student body's.

Larger-scale comparisons

More important than individual exam problems or homework assignments is the question, ***How do BioCS1 students perform in subsequent CS courses?*** Figure 4 shows the rate at which BioCS1 students choose to take CS2 relative to the student body as a whole. Our CS2 course covers data-structures and programming-paradigms and has not changed significantly in the past 15 years. It is a required course for computer science majors, but a significant number of other majors choose to take it because of computation's growing role across many disciplines. Figure 4 also shows the average CS2 grades on a four-point scale for each cohort.

Although none of the differences in Figure 4's data are significant except the rise in biology majors within the BioCS1 students, this experiment is one in which the absence of difference is, in fact significant. After all, the students in BioCS1 received half the number of lectures on core CS that the control and the full student body received. Even so, they chose to continue with CS at the same rate – and performed as well in those subsequent courses as both the control and the college population as a whole.

Control cohort

Of course, students who sign up for an experience like BioCS1 may have academic interests and priorities distinct from the entire student body! Thus, we also break out a control cohort in Figure 4's results. The control group comprises those students who, like BioCS1's students, indicated a desire and interest in taking BioCS1 but could not enroll because of space limitations. Those placements were decided randomly and established before any of the students had even arrived for their first semester at college. We realize that *not* getting a placement in a desired course may have had some difficult-to-measure repercussions. Although few educational experiments – particularly those requiring open buy-in from students – can afford the scaffolding of a true scientific control, we do feel fortunate to have at least this approximation of a control cohort against which to measure BioCS1's effects.

Affective outcomes

Our institution's formal end-of-course evaluations are required of all students in all courses: they provide useful insights into student perceptions of BioCS1. On a Likert scale of 1 (low) to 7 (high), students were asked to indicate their agreement with the statements (a) *This course stimulated my interest in the material* and (b) *I learned a lot*. The means from the Fall 2011 offering are summarized here for BioCS1, CS1, and all courses at our college, along with one standard deviation σ reported for the final column:

	BioCS1	CS1	All courses
(a) Stimulated my interest	6.55	6.80	5.64 (σ = 1.43)
(b) I learned a lot	6.73	6.81	5.86 (σ = 1.28)

Certainly, both BioCS1 and CS1 are perceived positively by students. In light of the CS venue, we do not consider the biology scores here. We were particularly heartened to see such similar scores between these early offerings of BioCS1 and the college's older, better-established CS1 curriculum. A second indicator of interest emerges from the rates at which students choose to take CS 2: Figure 4 shows that the rate was no worse (in fact better, though not significantly) for BioCS1 students. This difference has widened in our most recent cohort (2011-12): 17 of the 33 BioCS1 students (52%) have chosen to take CS2 course as their sole spring elective. In contrast, only 51 of the 134 CS1 students (38%) have chosen to continue with CS2 next term.

Workload and student impressions

As part of those end-of-semester questionnaires, both BioCS1 and CS1 students reported identical averages of 5.6 hours per week outside of class. The average over all courses at the institution is 5.5 hours (with σ = 3.4). We have been deliberate and careful to keep the workloads of the two courses as equable as possible, in order not to favor one over the other.

We are excited, too, that open-ended student feedback has echoed the positive results borne out by the statistics above. In 2011, for example, one of the anonymous comments commended the pairing: "I was really impressed with the amount of both CS and bio I could learn in one course and the integration of the two fields was very interesting. Alternating CS and Bio lectures worked well." Another explained in more depth *why* alternating lectures worked: "I thought the split between biology one day and the programming the other worked well. This is because one day we grasped the concepts of what we were going to do and the next we learned the tools to carry out the programs." Still others pointed to how the course affected them: "I thought I was going to hate CS, but now I'm planning to study it further" and "I wasn't considering a CS major; now I am."

Negative comments focused on schedule (labs were held on Friday afternoons) or opportunities lost: "I wish we did

Mandelbrot sets"; the control and full cohorts had done so. Yet these knock the college experience, not BioCS1 *per se*.

5. PERSPECTIVE

The BioCS1 course described here offers one way to integrate an introductory computer science course with an introductory biology course. The merger provides a rich and compelling context in which to teach computer science along with powerful computational tools with which to explore biological phenomena. The BioCS1 combination thus benefits teaching of *both* disciplines, despite the fact that the course necessarily omits some peripheral material from parallel CS1 and Biology1 curricula.

This result is a promising one for CS educators: after all, it demonstrates that CS1's context can be expanded to full-fledged cross-disciplinary collaborations without compromising the students' resulting computational skills and enthusiasm. Put another way, BioCS1 shows that a course with only half the usual number of lecture-hours can succeed as well as a traditional CS1 offering – or even better, if *applications* of computer science are an important part of an institution's or department's goals.

With respect to biology education, BioCS1 presents a field-tested curriculum through which students begin to develop a skill set in demand by labs and universities alike. As with CS1, we believe the slightly reduced content is more than made up by the interdisciplinary problem-solving skills that a hybrid course fosters. What's more, BioCS1 enables hands-on experimentation, albeit *in silico*, of advanced topics such as sequence alignment and phylogeny estimation; more traditional approaches necessarily abstract away the field's computational foundations.

Thus, we hope that this effort might spark parallel peer-as-context approaches in chemistry, engineering, mathematics, physics, and beyond. More generally, 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. COURSE MATERIALS

All course resources for BioCS1 including full lecture notes, assignments, labs, and capstone projects are available at [0].

7. ACKNOWLEDGMENTS

The authors thank HHMI award #52006301, NSF CPATH #0939149, and Harvey Mudd College for their generous support.

8. REFERENCES

[0] Dodds, Z., Libeskind-Hadas, R., and Bush, E. When CS 1 is biology 1: crossdisciplinary collaboration as CS context. ITiCSE '10, 219-223. Course URL: www.cs.hmc.edu/twiki/bin/view/CS6/

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

[2] Adams, J, Matheson, S, and Pruim, R. 2008. Blasted: integrating biology and computation. *JCSC* 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, *SIGCSE 35*: 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. 2003. 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. *JCSC* 18(5): 85-95.

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

[15] Parlante, N. et al.. 2010. Nifty assignments. In Proceedings, SIGCSE '10. ACM, New York, NY, USA, 478-479.

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

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

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

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

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

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

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.