Evolution of the 100 Problems Curriculum of Computer Science

Xiaoyan Hong[1], John C. Lusth[1], Nicholas A. Kraft[1], Debra M. McCallum[2]

Abstract – The 100 Problems (100P) approach to a Computer Science curriculum is formed around 100 concept- and research-related problems through nine core courses in normal computer science curriculum. It radically departs from traditional classroom-based learning formats and project based learning pedagogy in that it offers the entire curriculum with these problems, which, will guide students in their discovery of the subject areas. In the past two years, three cohorts of the students have been enrolled in the 100P program and have been studied. The students from the senior cohort are now approaching graduation. Our experiences offering courses in this format have been shaped from dealing with emerging challenges each semester. In this paper we will introduce the 100P program and describe our experiences and resulting strategies. We hope to offer a view that depicts the evolution path of the program towards the future.

Keywords: Guided discovery learning, problem-based learning, computer science education research.

I. INTRODUCTION

Classroom-based learning has been questioned as lacking a way to offer effective learning experiences to different types of learners. The National Academy of Engineering (NAE) has categorized learners into four types: highly self-motivated, self-motivated but needs coaching, motivated by external awards, and little motivated by goals imposed by others. While one way to address the problem is to adapt teaching methodology according to these categories, another more effective and efficient strategy is to encourage and help students to become highly self-motivated. The 100 Problems (100P) curriculum for computer science is created to teach with such a pedagogical method.

In short, 100P does not use traditional classroom-based learning formats, rather, it is formed around 100 or so concept- and research-related problems. The problems are structured to spread in nine core and elective courses in normal computer science curriculum, with additional mentor-assigned problems, and one final student-led problem. For each of the core and elective courses, students are given nine problems to solve for that semester. These problems guide students in their discovery of the subject area so to gain fundamental knowledge and skills. The students interact with and demonstrate the solutions to a mentor. Upon successful sequential demonstration of the solutions to a mentor and upon passing a final (oral) exam, a student is awarded credit for the course. Over their careers, the 100P students are freed from the bounds of the classroom and are empowered to teach themselves discipline-specific concepts, time management, and goal setting, all within a mentoring environment. Moreover, 100P students are required to demonstrate a deep and clear understanding of a concept before proceeding.

Starting Fall 2009, we have recruited four student groups (termed cohorts) in the 100P program from the CS0 class offered immediately before the recruiting semester. The students are recruited using two strategies. One strategy is to recruit from students who earned an A or a B. We used this strategy for the first (Alpha) cohort. The other is to recruit from all the students, which we used for the rest of the cohorts. As observed, though multiple students with a cumulative or major GPA of less than 3.0 have been included in the later cohorts, the retention rates do not show a bias towards the entry grade level. In fact, one of the top performers from the Beta cohort was the applicant with the lowest cumulative and major GPAs. The observation shows preliminary positive results of the 100P pedagogical method in strengthening the self-motivation and self-reliance in learning. Over time, we also observed a few issues to be addressed, which resulted in a few modifications to 100P in order to improve institutionalization.

[1] Department of Computer Science, The University of Alabama, Tuscaloosa, Alabama 35487, USA, {hxy,lusth,nkraft}@cs.ua.edu.
[2] Institute for Social Science Research, The University of Alabama, Tuscaloosa, Alabama 35487, USA, dmccallu@ua.edu.

One issue is the challenge to the 100P students in their time management [10]. All of the students who have dropped out of 100P have cited time management issues or lack of time as the reason for exiting the program. When students demonstrate time management problems all the time, the format of 100P may magnify the problem because early offered 100P courses have no deadlines for each problem. We have observed students starting slowly and attempting to finish several (or in some cases all) problems during the last two weeks of the semester. Although, Alpha and Beta cohort members demonstrated improved time management in their second semester, we developed time management guidelines in the 100P courses.

We also observed that some students preferred the 100P curriculum, however, they also demonstrated struggles in moving along with self-guided learning. One key cause is the weakness carried from their earlier CS0 course. To help the students to start 100P with a necessary background, we introduced an entry exam (qualifying exam) in our recruiting process. The qualifying exam aims at testing the concepts and also preparing the students for 100P. The administration of the qualifying exam reflects the pedagogical method of 100P.

Moreover, one major addition to 100P is a set of strategies to strengthen the self-motivation and self-reliance of the students. For most of the 100P courses we offered, we relaxed the nine-problems requirement, but added problems that help students to build a systematic knowledge body of the subject area without affecting the coverage of the concept guide for the course. Recently, we also require the students to develop concepts inventories as their final problem.

In this paper, we describe our modifications to 100P for the above issues. We also introduce two ways that a faculty can be involved in 100P. During the years, we have also received continuous feedback on the program through assessments. We will present some of the findings.

The paper is organized as follows. We first briefly introduce the 100P program in Section II. Then, in Sections III, IV, V and VI we discuss the issues regarding to prerequisites, building concepts inventory, time management and faculty involvement. In Section VII we report the evaluation results, and the related work is discussed in Section VIII. We summarize our work in Section IX.

II. 100P CURRICULUM OVERVIEW

For practical reasons, the 100P curriculum follows the normal curriculum. The 100P courses are given the same course number but with different section numbers. Thus the 100P courses can be handled easily within the current administration system. Over their careers, 100P students solve approximately 100 problems consisting of problems from 100P courses, mentor-assigned problems and a self-led problem. Among these problems, at least nine of the problems are to involve writing papers or presenting oral solutions, and at least 12 of the problems are to be group efforts. To satisfy ABET requirements, three of the writing and presentation problems focus on the History of Computing and three focus on Ethics. The mentor-assigned problems incorporate the mentor's research interests to better prepare the students for the future. The final problem to be solved is proposed by the student, with absolutely no restrictions attached.

Students are required to take 21 hours of core material in the 100P format. The core materials include: Computer Science I and II (CS1 and CS2), Data Structures (CS3), Algorithms and Complexity (AL), Operating Systems (OS), Programming Languages (PL), Automata Theory (AT), and Software Engineering (SE). Students are also required to take six hours of electives in the 100P format. The elective materials include: Artificial Intelligence (AI), Database Systems (DB), Computer Graphics (CG), Human Computer Interaction (HCI), Networks (NET), and Simulations (SIM). A 100P student may opt to take up to two of the core and elective CS courses in the regular classroom format.

Typically, for each problem, faculty will assign no-credit, half-credit, or full-credit. Full credit is given if a student demonstrates an acceptable solution within two attempts. Half credit is given if a student shows some grasp of the problem and solution but does not demonstrate an acceptable solution. No credit is given if a student does not demonstrate even a partially acceptable solution. The students are allowed two attempts of demonstration of

a problem to a faculty mentor. During a problem demonstration, the mentor interacts with the student to test the understanding of the concepts and the correctness of the solution. As a result, a student has a high chance earning full-credits and a grade of A to a 100p course.

100P's course web page typically posts links to the sets of current courses' problems, for example, the page for CS1 [11]. In addition, there is a link to the policies of the 100P program. This document explains how one enters and remains in the 100P program. It also describes how 100P courses are graded. It is important to note that if a student knows the procedures of 100P course and plan well, he/she can make multiple attempts at solving the course problems. The page usually includes links to the concept exam given towards the end of the semester and the materials for entry qualifying exam.

To achieve and improve upon the socialization that occurs in the regular classroom setting, 100P students are required to attend regular cohort meetings. These meetings are meant to build community, provide opportunities for peer mentoring, and encourage excellence (both individually and collectively). During these meetings, students discuss their difficulties, share their successes, teach other students about the interesting things they have learned, and participate in team-building activities and projects (including service learning activities/projects).

III. PREPARATION AND RECRUITING

The need for an entry qualifying test merged after two years of the program when the teachers observed a few cases in which the problems were passed with excessive mentor efforts and time. The reasons appeared to be centered on the prerequisites to 100P and the transition on learning style. The premise is that the students show strong motivation for enrolling in or continuing the 100P program. The 100P program thus developed policies helping them to overcome the weakness and to prepare for further success. The policies follow the spirit of 100P, i.e., help students to become highly self-motivated and self-resilience.

A. Entry qualifying exam

One policy is to institute qualifying exams for entering 100P. The prerequisites for entering 100P are publicized as the concepts inventories. The questions in the concepts inventories cover the material from the CS0 or CS1 courses in associated with the two entry points of 100P. The actual exam will consist of a randomly selected subset of questions, which may be tweaked from their appearances in the public set. A student is offered a link to the set of sample qualifying exam questions. A student must score 90% or better in this exam to pass the qualifying exam. Currently, there are two entry points for the students to enroll in 100P program. If students enter after taking the course of CS0, they must pass qualifying exam on CS0 concepts inventory. The high level entry point is after students taking CS1 in the traditional class-room based format. The interested students must pass the qualifier testing based on CS1 concepts inventory.

Usually a qualifying exam will be given one day before the start of a semester for an organized group exam. A student may take the qualifying exam more than once, if needed. A student must succeed in passing the exam by the last day to add classes in the academic calendar (usually one or two weeks into a semester). Since a 100P course is offered with the same regular computer science course number with a different section, a student must attend the regular classroom-based section of the course (either CS1 or CS2) until they pass the exam.

The 100P curriculum is advertised and students are invited through the course of CS0, which is the first curse for undergraduate majored on CS. In CS0, students are given sample problems to solve so that they may gain a sense of whether the 100P program may be for them. The faculty also make it clear that the program is designed to bring out the students' maximum potential through a team-based, problem-solving approach. Before a semester starts, with enough leading time so that the students will be able to register classes, an invitation email will send to students who have taken CS0 or CS1.

B. Sample qualifying questions

Two sample qualifying questions are give in Fig. 1. Both are on the major concept of **Loops**, picking from a set of questions testing the subconcepts of **Loops**. One tests the subconcept on counts and accumulations, and other tests the subconcept of picking patterns, counts and accumulate.

Concept: recognizing *counts* and *accumulations*
- Which pattern does the following function implement?

```
def f(items):
    m = 0
    for i in range(0,len(items),1):
        m = m + items[i]
    return m
```
  1) the *filtered-accumulate* pattern
  2) the *filtered-counting* pattern
  3) the *counting* pattern
  4) the *accumulate* pattern

Concept: picking patterns, *counts* and *accumulate*
- If I wish to solve this problem: *determine the sum of the numbers from* a *to* b, I should implement the:
  1) the *counting* pattern
  2) the *accumulate* pattern
  3) the *filtered-accumulate* pattern
  4) the *filtered-counting* pattern

Fig. 1. Testing two subconcepts relating to major concept *loop*

IV. GROWING CONCEPT INVENTORY

A course in the 100 problems (100P) format comprises a concept guide, a problem set, and a comprehensive oral exam. The problems for the course and questions for the oral demonstrations of the problems and the final oral concepts exam are constructed to cover the concepts in the concept guide. Our concept guides for CS2 and data structure are reported in [10].

Each problem may cover more than one concept, and one concept may be covered by one or more problems. Similarly, a concept exam question may cover more than one concept, and each concept is covered by at least one question. In addition, we now offer a list of recommended materials about the concepts covered on each problem. Usually, the recommended materials include textbooks, faculty developed writeups, public postings such as Wikipedia's takes. A typical problem includes major concepts, concepts relating to the major concepts and suggested readings, tasks to perform for the problem, and additional requirements for demonstrating the problem to a mentor.

A. Sample problems

The very first problem that the 100P students will take is given in Fig. 2 as an example. The problem is from the 100P course CS1 from Fall 2011 offering. The problem has very detailed instructions to guide the students for the problem requirements and also prepares them for the 100P style. In addition, we show a higher level 100P course, *Programming Languages*, in Fig. 3. The list of problems and one example of the problems are given in the figure. In this course, the problem descriptions are written in high level and the mentor has more opportunities to interact with the students to test their understanding of the concepts.

B. Building systematic view of concepts

100P program promotes independent study on the course subject areas. However, students need more direction in obtaining systematic knowledge although many concepts can be obtained from books and from on-line. Gradually, we have developed a few ways to help students to navigate through all the information sources. In Fall 2010, we started assigning a textbook for a course and requiring the students to create a study plan based on the course

concept guide and textbook used. The assignment is given as a group based problem. In Spring 2011, we were able to add faculty self-developed materials. And when course offering continues, the self-developed contents will be enriched. In Fall 2011, we started to request students to build own concepts inventories through the format of final exam. These activities not only help the students to build a knowledge system of the subject areas but also help the program to sustain in long-term.

V. TIME MANAGEMENT

100P allows students to work at a pace they set (within the constraints of the semester structure). However, insufficient time management skill has become a major stress to the students. It also created stress to the faculty when many of them tended to demo many of their solutions during the weeks approaching the end of a semester. This also left students less time and opportunities to revise their work and re-demo for passing. While encouraging them to work on problems early, the current 100P courses use deadlines for the assignments.

The management of deadlines for problems is subject to the students and their instructor for a specific course. One key component to consider is to give time for re-demonstrating a partially passed problem. In the past, we have tried to give a grace period for re-demo after a hard deadline of first trial, or to impose a hard deadline for the final pass and to let the students to manage their time. A deadline can also be given to a problem or a group of problems. Our experience shows that students are more likely to underestimate the time needed for working out a problem, that'll result in rescheduling a demo time slot, or more re-demo cases.

To encourage more timely submission of problems and also to set at a student's pace, two new schemes have been added for 100P course management and policy. One scheme is to request each student to submit to the faculty member in charge for a 100P section a schedule of demonstrations. This schedule is due one week after the last day to add classes and will list deadlines for successfully demonstrating completion of the problem tasks assigned (i.e., Problem X successfully demonstrated by the end of the week Y). Acceptance of the schedule by the faculty member followed by a failure to adhere to the submitted schedule will result in a lowering of the course grade. An acceptable schedule will spread problem demonstrations evenly throughout the semester and will also allow time for multiple demonstrations of a single solution should an initial demonstration uncover disqualifying flaws.

We further developed a new grading scheme in associated with deadlines. Currently each missed deadline with respect to the submitted schedule will result in a drop of one-third of a letter grade, provided the problem solution is successfully re-demoed within a specified grace period. For example, one missed deadline would reduce an A to an A^-, while two missed deadlines would reduce an A to a B^+. There is no penalty for re-demoing before the deadline. The grace period is set by each instructor, but is typically 48 hours. Failure to successfully demo by the end of the grace period will result in an unacceptable problem solution.

VI. FACULTY PARTICIPATION

Over the time, 100P program is constantly challenged by faculty participation. A faculty work load for a 100P section with 10 - 12 students is equivalent in time to a three-hours lecture based class. And there is an advantage of no grading outside of the meeting with students. However, depending on the number of re-demos, the work load can increase. Current implementation of 100P relies on whether a faculty is available for supervising a 100P section of a regular computer science course. Our experience suggests that there are different ways that a faculty can participate in 100P program. For lower level CS courses, such as CS1 (CS260) more individuals are able to mentor a student.

1) method 1: adopt a problem: Faculty participation can consists of picking a problem and having some or all of the 100P students demonstrate their solutions to one faculty. Take an example of CS1 [11], the faculty's job is to ask questions about their strategies for implementation a problem and the runtime complexities of their choices. If a student can answer faculty's questions satisfactorily and he/she has made the proper implementation choices, the student is given a pass. Otherwise, the faculty will tell the student what to fix and the student has one chance to

come back and re-demo. If the student does so successfully, he will earn a pass. If not, the student is given a fail or a half-pass, depending on the faculty's judgement.

2) method 2: adopt a student: To serve as a 100P mentor at a student-base, the faculty will mentor the same set of students throughout the semester. A faculty usually allocates one hour per student per problem to meet with and go over 100P problem solutions. There are usually seven or eight problems, but some times, especially at the beginning of the semester, students have to make corrections and re-demonstrate their revised solutions. For each problem, faculty mentors are provided a list of questions to ask and tasks for their students to perform to ascertain their knowledge of concepts and code.

VII. Evaluations and Findings

Both quantitative and qualitative data have been collected for assessment through the years. The quantitative data is obtained through questionnaires to the students. Qualitative feedback was collected through focus groups as a supplement to the data collected through student surveys. During the second year of the program, electronic collection through the use of email and Survey Monkey is also used. This method provided program implementers with feedback from students even as the number of students served by the 100P program expanded.

A. Student surveys

Two questionnaires are prepared for obtaining quantitative data. One is general course evaluation in many aspects, and the other is on each problem in an offered course. Table I shows the mean values for the questionnaire items from the general evaluation questionnaire. The picked items relate to the challenges discussed in this paper and also the perceptions that might change over time (i.e., they are not related to instructor characteristics, or the specific course).

In terms of overall ratings of the courses (e.g, items $6, 7, 8, 9, 13, 14, 15$), the students were more likely to say the format was well-suited for their learning style, yet they were also more likely to say they would prefer to have a lecture before each type of problem. But this preference showed a declining trend when they took more 100P courses. They all valued the learning experiences. There is a lower value for both Alpha and Beta cohorts in Fall 2010, that was the semester that both cohorts were taking two 100P format courses. Nevertheless, they were more likely to say that they'd prefer 100P format over traditional lecture and would recommend 100P to others. However, we also found there was a declining trend when the students move to a higher level course. A few factors could contribute to the decline, e.g, the nature of the senior level courses, more-than-one 100P courses, smaller sample size, etc.

Compared to other non-100P courses, the students responded with mean values above the mid-point of the scale, and in most cases they are above 4.0 on the 5-point scale (see items $6, 12, 13, 14$). Those items include the amount of learning compared with other courses, feeling well-prepared for next course, a valuable learning experience, and recommendation for course in 100P format. Because the 100P program has such a different grading structure, it is difficult to actually compare the grades in the 100p courses with the grades in the same non-100p courses.

In addition, there are a few interesting observations. One observation is that the ratings were less positive for both Alpha and Beta cohorts in Fall 2010. In that semester, both cohorts were taking two 100P format courses. Another observation is a little conflict in their answers relating to deadline and pacing. While they were more likely to say they had trouble pacing themselves and getting enough work done, they were less likely to say they would prefer to have deadlines for problems. Since some of the data were collected in early semesters, we have incorporated new strategies on in later 100P course offerings, deadline is an example. These data and findings also suggest that new assessment questions should be planned in the future to obtain their opinions on these mentioned issues.

TABLE I
EVALUATION RATINGS

Questions	Alpha				Beta			Gamma	
	F09	S10	F10	S11	S10	F10	S11	F10	S11
1. Background prepared me to take this course[1]	4.00	4.75	4.13	4.33	4.00	3.75	4.00	3.50	4.00
2. How often did you seek help from faculty mentors?[2]	2.78	2.62	2.50	3.00	3.00	2.19	1.6	2.67	3.00
3. How often did you seek help from other students in the class?[2]	3.90	3.75	3.38	4.00	4.14	3.75	3.2	3.33	3.75
4. Town Hall (group) meetings were effective use of time[1]	4.20	3.88	2.75	3.33	4.14	3.37	3.20	3.17	3.75
5. Amount of time required compared with other courses in major[3]	4.00	3.50	3.12	4.00	3.86	3.00	5.00	3.83	4.75
6. Amount of learning compared with other courses in major[3]	4.20	4.57	3.12	4.00	4.43	3.63	4.00	4.17	4.50
7. Format well-suited for learning style[1]	4.40	4.43	3.25	4.00	4.43	4.06	3.60	3.83	3.75
8. Prefer 100P format over traditional lecture[1]	4.80	4.88	3.38	4.00	4.86	4.31	3.00	3.67	3.75
9. Would prefer to have lecture before each type of problem[1]	3.80	3.38	4.00	3.5	2.71	3.69	3.4	3.33	4.00
10. Would prefer to have deadlines for problems[1]	2.70	1.50	3.63	3.00	2.14	2.81	3.00	2.83	3.75
11. Had trouble pacing and getting enough work done[1]	3.60	2.88	3.75	2.50	2.86	3.13	3.60	3.33	3.50
12. Feel well prepared for next CS course[1]	4.60	4.50	3.50	4.50	4.57	3.63	4.00	4.17	4.00
13. Course was valuable learning experience[1]	4.80	4.75	3.50	4.50	5.00	3.63	4.40	4.67	4.50
14. Would recommend course in 100P format to others[1]	4.70	4.75	3.38	4.00	5.00	3.69	3.40	4.17	4.25
15. Plan to stay in 100P program (% saying "yes")[1]	100%	100%	75%	100%	100%	87.5%	60%	66.7%	100%

[♮] Items were answered on a scale from 1-5, with the following endpoints:
[1] Strongly disagree - Strongly agree; [2] Never - Very often (at least once a week); [3] A lot less - A lot more.

B. Focus group study

During the first two years of the program, three focus group meetings were held with participating students. The goal of these meetings was to provide the implementers of the program with knowledge regarding how the program was perceived and received by the students as the curriculum was developed, implemented and refined. Students in each of the three cohorts responded by providing their "take" on the open-ended questions regarding their experiences with the 100P program. In the later focus group meetings, the students also responded to those issues that had been addressed during the previously-held focus groups.

In terms of the 100P style, the first cohort of students (Alpha cohort) reported in the first focus group meeting that the greatest challenge was a tendency towards procrastination on their parts. However, all students felt that they had improved in this area over the course of the semester. They also reported struggling with solving the required problems at first but felt that they eventually developed strategies (searching Google and Wikipedia; asking for lectures from instructors when needed; consulting one another) that helped make the work easier. In the second focus group which was held after their second semester, the entire cohort felt strongly that they would all prefer to stick with 100P format courses through graduation and would recommend the program to other self-motivated students. Feedback from all the cohorts in the last focus group study echoed the previous results. Again, the independence factor of the program was mentioned by many participants as both a pro and a con. Students found some frustration with the need to be self-motivated and manage their own time appropriately; however, they also saw the value in the independent research that was required to solve the necessary problems on their own and still enjoyed the flexible deadlines.

In evaluating how much they have learned, the students regularly mentioned that the 100P format provides a greater depth of learning as opposed to the greater breadth offered in traditional course format. All participants reported feeling that they are "learning how to learn" and gaining valuable strategies for working independently. Participants felt that the 100P curriculum was perhaps more difficult than the traditional format but was a fair alternative and particularly helpful in providing them with the opportunity to learn to work more independently and gain more in depth knowledge about problem-related topics. All students reported gains in self-motivation and self-regulation during each semester. A remark on this point from the authors is that this is the true advantage of 100P. Because a student is required to successfully demonstrate each problem with interaction with the mentor

during the demonstration and one chance of re-demonstration. Such a process reenforced one's learning and also gave the student a chance to earn a grade A for the problem.

Talking about the time management skills, students recognized and recommended on setting deadlines for themselves and being prepared to stick to them. They later reported that their time management skills had seen even greater improvement and that working together had become even more productive due to previous experience. Some notable new comments included the students' appreciation for how helpful the flexibility of deadlines for 100P courses was when working around deadlines of other courses and that more resources (scanner/copier, other textbooks) would be useful.

Another feature of 100P is its social structure. Related to this, students reported feeling as if they had more interactions with each other than a regular class would require. The small size of the group and the ability to attend "Town Hall" meetings together were endorsed by everyone as very helpful. Group problems are generally preferred to individual problems by everyone.

In summary, overall evaluation ratings for the program are high and the students in the program appear to be satisfied. The most consistent request by participants was for a bit more solid "direction." The "Big Book" was mentioned as a helpful resource. Most students mentioned a desire for at least an occasional lecture and/or a more "concrete" resource which they could reference when their preferred resources (such as the internet) presented inconsistent information. Look ahead, the Alpha cohort, which is involved in the 100P program over the longest period and is also approaching graduation, reported increasing gains in the above areas during each of their four semesters with the program.

Future assessments can add instrumentations to compare the performance of the 100P students and non-100p students. Potential methods could be standard tests, grades when both groups taking the same non-100P course, etc. Another interesting area is to study how the male and female students percept the 100P approach. Currently we are only able to say from the focus group discussions that the females are just as enthusiastic as the males in liking the program. But there are very few females in the 100p program, this issue would be investigate through a long time period.

VIII. RELATED WORK

Chubin et al. [3] report that in the engineering work force, diversity includes not only race, ethnicity, and gender, but also learning style, world view, and problem-solving orientation. To this end Felder and Brent [5] explore student differences, including different learning styles, different study orientations, different approaches of learning, and different levels of intellectual development. Many teaching strategies and methods that have been adjusted to learning styles have been studied and have been found to effectively induce a process of deep learning among students. These strategies and methods can be used to address the urgent need for STEM undergraduates to experience inquiry-based learning [12] and include problem-based learning [1], project-based learning, active learning, and cooperative learning. 100P incorporates aspects of these strategies and methods to encourage and foster deep learning among students; these aspects are selected with the goal of addressing the fundamental problem of "learning to learn."

Problem-based learning (PBL) has shown success in increasing student learning [8]. 100P uses PBL as its primary vehicle. However, rather than use a single overarching problem for an entire course, 100P uses a number of problems that can be tailored to the interests of the participating students and faculty. The 100P approach bears some resemblance to Becker's successful use of inquiry-based learning (IBL) to replace the traditional lecture-based format in an introductory Computer Science course [2]. 100P provides more guidance in that students do not necessarily pick and choose problems. Furthermore, the scoring method of Becker's approach is replaced by requiring all submissions to be "exemplary". 100P bears a strong resemblance to studio-based courses of Gonsalvez and Atchison [6]. In two IT courses, students work on a variety of structured and unstructured tasks, with formal teaching worked in as appropriate. Docherty et al. [4] take the studio approach further by integrating it throughout the entire curriculum of an IT degree. The studio classes described in both of the previous papers supplement regular IT or CS courses and students work in teams. 100P extends the studio idea to full CS courses while

balancing individual and team efforts. 100P also has a strong relationship with recent work in studio-based learning by researchers at Auburn University, the University of Hawaii, and Washington State University [9], [7]. However, the studio-based approach retains lectures, while 100P is forgoes the classroom entirely.

IX. CONCLUSIONS

100P program has demonstrated advantages due to its problem-based approach and guided discovery pedagogy. This paper introduces some new practices and policies to reflect the current realities of 100P and suggestions on faculty involvements. Overall, students felt it is a valuable learning experience and they learned more in depth through the "process" of problem solving than they would have from lectures. The students get much better feedback on their work product. They grow to be more self-motivated and self-resilient in learning. They also form a closer relationship with their teachers. The weaker students, if they did not drop out, showed great gains in ability. The students really seem to expect more opportunities for taking courses in 100P format. The 100P program has been a success and it is also a great recruiting tool.

ACKNOWLEDGMENTS

REFERENCES

[1] Problem-Based Learning: An Introduction. *The National Teaching & Learning Forum*, 8(1), 1998.
[2] K. Becker. Reconciling a traditional syllabus with an inquiry-based introductory course. *J. Comput. Small Coll.*, 20(2):28–37, 2004.
[3] D. E. Chubin, G. S. May, and E. L. Babco. Diversifying the Engineering Workforce. *Journal of Engineering Education*, 94(1):73–86, Jan. 2005.
[4] M. Docherty, P. Sutton, M. Brereton, and S. Kaplan. An innovative design and studio-based CS degree. *SIGCSE Bull.*, 33(1):233–237, 2001.
[5] R. M. Felder and R. Brent. Understanding Student Differences. *Journal of Engineering Education*, 94(1):57–72, Jan. 2005.
[6] C. Gonsalvez and M. Atchison. Implementing studios for experiential learning. In *Proceedings of the Australasian conference on Computing education*, pages 116–123, 2000.
[7] D. Hendrix, L. Myneni, N. Narayanan, and M. Ross. Implementing studio-based learning in CS2. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 505–509, 2010.
[8] C. E. Hmelo-Silver, R. G. Duncan, and C. A. Chinn. Scaffolding and Achievement in Problem-Based and Inquiry Learning: A Response to Kirschner, Sweller, and Clark. *Educational Psychologist*, 42(2):99–107, 2006.
[9] C. Hundhausen, N. Narayanan, and M. Crosby. Exploring studio-based instructional models for computing education. *SIGCSE Bulletin*, 40(1):392–396, 2008.
[10] N. Kraft, X. Hong, J. C. Lusth, and D. McCallum. Experiences with CS2 and Data Structures in the 100 Problems Format. In *Proceedings of the IEEE 2011 Frontiers in Education Conference (FIE)*. IEEE, Oct, 2011.
[11] looP Program. Introduction to Computer Science (100P version)–Problem Set. http://beastie.cs.ua.edu/100P/CS1/, 2011.
[12] Project Kaleidoscope. Recommendations for urgent action. Technical report, 2006.

Xiaoyan Hong Dr. Xiaoyan Hong is an associate professor in the Department of Computer Science at the University of Alabama, USA. She received her PhD degree in Computer Science from the University of California, Los Angeles in 2003. Her research interests include mobile and wireless networks, future wireless Internet, vehicle networks and mobility modeling.

John C. Lusth Dr. John C. Lusth is an associate professor in the Department of Computer Science at the University of Alabama, USA. Before joining Alabama, he worked at Boise State University and the University of Arkansas as an assistant professor and associate professor respectively. He received his PhD degree in Computer Science from the University of Alabama in 1997.

Nicholas A. Kraft Dr. Nicholas A. Kraft is an assistant professor in the Department of Computer Science at the University of Alabama, USA. He received PhD degree in Computer Science from Clemson University in 2007. His research is in the areas of software maintenance and evolution, program comprehension, mining of software repositories and software language engineering.

Debra M. McCallum Dr. Debra M. McCallum is the director and a senior research scientist of the Institute for Social Science Research at the University of Alabama, USA. She received her PhD degree in Social Psychology from the University of North Carolina, Chapel Hill in 1980.

Problem 1: Arrays and Order Notation
Readings

- *An introduction to Order Notation*
- *An introduction to ADTs and Public Interfaces*
- *Arrays*
- *Fillable Arrays*
- *Circular Arrays*
- *Dynamic Arrays*
- *Dynamic Circular Arrays*
- *Wikipedia's take on Dynamic Arrays*

Tasks

- *order notation*
- *arrays*

Order notation task

- *Be able to answer every question and any related variation of the questions on the class concept inventory that relate to order notation.*

Arrays task

- *Implement the following classes with these operations:*
- *a circular array class*
    - *insert-at-front*
    - *insert-at-back*
    - *insert-at-index*
    - *remove-from-index*
    - *remove-from-front*
    - *remove-from-back*
- *a dynamic array class ... (cut in this paper)*
- *a dynamic circular array class*
    - *insert-at-front*
    - *insert-at-back*
    - *remove-from-front*
    - *remove-from-back*
    - *grow*
    - *shrink-to-fit*
    - *extract*
    - *find*

In addition to the operations listed above, all classes should implement is-empty *and* is-full *(if appropriate) as well as* display*.*

Each class should be implemented in a separate file. Provide a main function that tests the interfaces of each class in a way that your Mentor can easily ascertain the correctness of your solution.

Be able to describe the complexity of each operation in the public interface of your classes using order notation.

Note: any operation should be implemented as efficiently (order-notation-wise) as possible. Do not spend time optimizing an operation unless directed.

Fig. 2.   Problem 1 from Fall 2011 offering of CS2.

Introduction
Here are the problems and readings for the class. Make sure you understand the readings before attempting a problem and demonstrating your solution. The suggested readings are only a start; you should continue your education by reading other materials.

Problem 1: Functional programming
- *Problem Description*

Problem 2: Lexing
- *Introduction to implementing programming languages*
- *Introduction to lexical analysis*
- *More on lexical analysis*
- *Problem Description*

Problem 3: Functional Programming II
- *Problem Description*

Problem 4, 5, and 6: Parsing and Pretty Printing
- *Introduction to grammars*
- *LL(1) grammars for Associativity and Precedence*
- *An introduction to language recognition*
- *An introduction to parse trees*
- *Problem Description*

Problem 7: Fun with Programming
- *Problem Description*

Problems 8 and 9: Programming Language Interpreters
- *Environments*
- *Problem Description*

Oral exam preparation
Here are two documents to prepare you for the oral exam:
- *concepts*
- *possible tasks*

Example: Problems 4, 5, and 6
Topics to learn
Please read up on the following subjects:
- *grammars*
- *recursive descent parsing*
- *building parse trees*
- *pretty printing*

Tasks to perform
Your task is to design a programming language of your own design and write a pretty printer *for your language. A pretty printer starts with a recognizer; a recognizer determines whether or not the lexemes are in the proper order. Next, the parser is augmented to build a parse tree out of the lexemes it has recognized. This parse tree is then passed to a pretty printer which recreates the input file.*
A good approach to solving this problem is to build the recognizer and pretty printer in parallel. That is to say, implement a recognizer for your lowest level grammar rule. Once that appears to be working, modify the parser to build a parse tree for that rule and create a pretty printer that prints the parse tree. Once this is all debugged, add another grammar rule to your parser and pretty printer, and so on.
You may use any language for this task.
This counts as three problems.

Fig. 3.   Problem list from Fall 2010 offering of CS403 Programming Languages. Each problem is linked to a file with details.