CQI Report
CS Fundamentals
Fall 2018

Summary

The CS fundamentals committee met a few times each semester to ensure that information flows from one course to the next, that instructors at different levels of CS1, 2, or 3, have the ability to share their instructional methods / new approaches, and the concerns they may have about students' retention of skills moving from one course to the next. Overall, all courses in the fundamentals' sequence satisfy their outcomes. Nevertheless, some changes are recommended as outlined below.

General Recommendations

Although the outcomes are achieved (to varied degrees), we recognize that the granularity of details of our outcomes sometimes hinders our ability to fully diagnose our students' ability. We recommend looking at the outcomes and identify how to address this problem.

In addition, we recommend ensuring a better communication with EPCC so that any changes in instruction be discussed and reflected in both institutions. A meeting a semester should take place.

Below are more targeted recommendations for each of the CS1, 2, and 3 courses, followed by each individual report.

CS1301/CS1101: Introduction to Computer Science

Summary of Observations and Recommendations

Coverage of topics early in the semester.

Based on the results of outcomes coverage considering grades and attendance, it is recommended to cover most of the topics early in the semester and before the last three weeks of the semester. Time is provided throughout the semester to reviews and more practice that could be left for the last three weeks, with more independent work on topics that need reinforcement.

Growing enrollment.

Facing a growing enrollment, in addition to larger lecture sections, we now also have larger lab sections. This is manageable, but we found it useful to have 2 TAs in a large lab (not just one), along with a PL and an IA. Having IAs and peer-leaders in addition to TAs was instrumental in providing better assistance to students. The IA assigned to the course was in charge of assisting students during the lab (a large lab with more than 45 students) and also reaching out to students missing classes or with low grades. Personal attention made a difference to those students. Recommendation: Having 1 TA (with 1 PL and 1 IA, or either) per smaller labs, with more lab sections, would be ideal. In case it cannot be done, we recommend having 2 TAs in large labs (45+ students).

Instructional team.

TAs, PLs, and IAs are crucial in the success of our students. Their skills are essential in delivering relevant content in labs and in providing relevant and timely guidance. It is important to make sure that they are all properly prepared: they should be knowledgeable in the programming languages taught in this course, but also familiar with the goal of the course and trained in pedagogy. In particular, giving them some formal training on speaking to large classes (ensuring to broadcast voice, move about the room, etc.) and general teaching paradigms could be very helpful.

Training that was put in place allowed us to have very good TAs and instructional team members in general, but we need to stress how important this training is so that it continues. Additionally, we recommend creating a pipeline of instructional assistants (IA, PL, TA) so that we ensure the continuity of quality of instruction semester after semester.

Lab and Lecture with same instructor

The lecture and labs of CS1 have several sections. It would be very advantageous to students if we could guarantee they had a matching lecture to lab, so we can ensure continuity of education. Students attending both lecture and lab with the same instructor are effectively covering the same topics in lectures and labs sessions. Students in different sections may see topics at different times, which may be a disadvantage to some students. Students in this situation have communicated this concern verbally and it is recommended to make efforts to enroll students in the two courses under the same instructor.

Teamwork

Having students in teams to perform coding (during the lab session) tasks seemed engaging, beneficial and helpful. Even though these activities were not graded, it helped solidify concepts and have students who did understand help those who might have not understood.

Continuous evaluation

In addition to exams, homework and lab assignments, adding a weekly survey and quiz for the lab (as it was done for the class/lecture) was beneficial to have a sense of the understanding of the topic being introduce before moving into the next one. It is recommended to have continuous feedback from students, which can be done in Blackboard to address the increasingly large groups observed in CS1.

Academic dishonesty

A case of academic dishonesty involving plagiarism of code found online was found and reported to the Office of Student Conduct and Conflict Resolution and was handled properly by TA and promptly by the office. This incident happened towards the last half of the course. Lessons learned from this are that students should be provided with specific examples and reminders throughout the course as per the meaning of plagiarism and collusion. The problem found online was one not applied in the context of UTEP as other problems and it was a generic one that could be found. It is recommended to use new problems that cannot be found online.

CS2401: Elementary Data Structures

Summary of Observations and Recommendations

1. Based on the third section of CS2401 experience in fall 2018, we recommend to try and accommodate struggling students in smaller groups and see if this increases their retention and success consistently over time.
2. We recommend to keep using an online textbook with reading and homework tracking.
3. We suggest adopting a way of assessing students based on skill acquisition since it is more attuned to our students' life circumstances (heavy work load, children, family commitments, etc.).
4. We propose to reconsider the sequence in which topics are introduced in this course to make it flow better for students (to be tested in spring 2019).

More specifically:
5. We recommend removal of outcome 2.5c to level 1. The topic is covered in the next course.
6. While the dropping and failing rates are typical, there is a scope of improvement by reducing these rates. A recommendation is to check if there is a correlation between the dropped or failed students with their grades in earlier courses in Computer Science. Possible interventions include mandatory tutoring for students and membership in programming clubs who do not do well at the beginning of the semester. TAs, peer leaders, and the Head TA identified the students at risk in the Fall of 2018 semester but not all the students who ended up with D and F took the advantage of the additional support the department provided. Counseling from the department about challenges of the struggling students is another possible direction.
7. We acknowledge the quality of the TA training and recommend the TA organization with a head TA be continued. We only suggest to focus more of the TA training on time management so that they can do well both in teaching and in their own studies.

CS2302: Data Structures

Summary of Observations and Recommendations

- The program that supports instructional assistant should continue to be funded. Students benefited from the IA greatly.
- We can do a much better job with code reviews. Given them an official rubric and assessing them more thoroughly is necessary for the idea to work.
- Students actually read the zyBook, we should continue using it.
- Even though not a lot of students dropped the course, many of them stopped showing up. Something must be done.
- Having multiple versions of a lab is great for students, but it's time-consuming for the instructor. We should think about how to proceed.

INDIVIDUAL COURSE REPORTS

1. CS1301/1101: 2 lectures and 1 lab
2. CS1301/1101: 1 lecture and 1 lab
3. CS1301/1101: 1 lecture and 1 lab
4. CS2401: 2 sections, including lab
5. CS2401: 1 section, including lab
6. CS2302: 2 lecture sections

CS 1301/1101 Introduction to Computer Science
2 lecture sections and 1 lab section
CQI Course Review – Fall 2018
Martine Ceberio

1. Course Description

Students in this first course for majors in Computer Science will learn to be active learners, understand the motivations for computing, basic concepts of algorithms, basic computer organization, and impacts of computing. They will develop problem-solving skills, implement solutions to computing problems in a high-level programming language, and build team skills, critical-thinking skills, and professionalism.

Knowledge and Abilities Required Before the Students Enter the Course: Students entering the course are not required to have a background in Computer Science or programming. They should be familiar with topics from Pre-calculus, including algebraic functions, proofs, and base representations of numbers.

Prerequisite: MATH 1508 or MATH 1411 with a grade of C or better.

Textbook

Online text: Programming in Java, zybooks, available at zybooks.zyante.com.

2. Learning Outcomes

Level 1: Knowledge and Comprehension: Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to describe, at a high level:

1. The history of computing
2. The relation between computing and society, including main social, ethical, and legal issues
3. Computing as a profession, from required knowledge and skills to major career options
4. Computer representation of simple data types and operations, including operations with binary numbers
5. Differences among programming languages
6. Pseudocode of the use of Multi-D arrays
7. Pseudocode of the use of Linked-Lists

Level 2: Application and Analysis: Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able:

1. To analyze problems and express solution algorithms in pseudocode, including a correct use of:
    a. Arithmetic and logical expressions
    b. Simple I/O operations
    c. User-defined subprograms, including recursive methods
    d. User-defined types
2. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults
3. Use teamwork roles and methods in the classroom

Level 3: Synthesis and Evaluation: Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course, students will

be able to use the syntax and semantics of a higher-level language to express solutions to programming problems, including the correct use of:
1. Basic variable types such as integer, real number, character, string, 1-D array
2. Assignment, arithmetic, and logical operations
3. Basic control structures: if-then, for-loop, while-loop

3. Course Approach

This course is usually taught as follows: most of the topics are covered early in the semester (first 4 weeks) and are covered again in more depth before the last three weeks of the semester. Time is then provided throughout the end of the semester to review and offer more practice.

In this course in fall 2018, a few modifications were made. Based on the Google Faculty in Residence program, a few elements of the course were made more salient: namely testing and problem solving. Testing was introduced in the first 3 weeks of the course and used throughout the semester (since the notion had been introduced, it was easy to refer to it over and over again for increased practice and understanding). Problem solving was brought as an off-line activity once a week in lab (30 minutes a week), when teams of students (with roles assigned from oracle, to problem solver, to tester, reviewer) had to solve a problem on the board and then report on their performance.

In addition, I tried to change the culture of the course by normalizing failure, sharing our backgrounds and aspirations, through ice-breaking and life-path activities early on during the semester.

4. Observations and Instruments Mapping to Outcomes

CS1301 Section 1:

There were 43 students originally registered for the class at the time of the first day of classes: 6 only were female students (14%).

- Two of these students never showed up;
- 4 had to drop for reasons outside of school (sickness, work schedule, family commitments): 3 of these were women
- Out of the 37 students remaining (then down to 8% of women left),
    - 7 dropped (19%), including one woman, during the semester;
- Out of the 30 remaining,
    - 24 passed the class (80%): 12 with an A, 8 with a B, 4 with a C
    - Out of the 3 female students: 2 passed the class and one failed, with1A, 1B, 1D.

Grades	A	B	C	D	F
All	12	8	4	4	2
Female Students	1	1	0	1	0

CS1301 Section 2:

There were 31 students originally registered for the class at the time of the first day of classes: 7 only were female students (22.5%).

- Three of these students never showed up, including one female student;
- 1 had to drop after a serious car accident that occurred at midterm;

- Out of the 27 students remaining (then with 22.2% of women left),
    - 2 dropped (7%);
- Out of the 25 remaining,
    - 19 passed the class (79%): 7 with an A, 8 with a B, 4 with a C
    - Out of the 6 female students: 4 passed the class and 2 failed, with 2B's, 2C's, 1D, and 1F.

In both lecture sections, students were identically evaluated throughout the semester via quizzes (mostly online), bi-weekly homework assignments, three mid-term exams and a comprehensive final exam. Students were also evaluated based on their attendance and participation in class.

Grades	A	B	C	D	F
All	7	8	4	3	3
Female Students	0	2	2	1	1

CS1101 Lab section:

There were 55 students originally registered for this lab at the time of the first day of classes: 7 only were female students (18%).

- 4 of these students never showed up;
- 4 had to drop for reasons outside of CS (family, sickness, work schedule, etc.), including 3 women;
- Out of the 47 students remaining (then with 14.8% of women left),
    - 3 dropped (6.4%);
- Out of the 44 remaining,
    - 29 passed the lab (66%): 9 with an A, 12 with a B, 8 with a C
    - Out of the 7 female students remaining in the lab: 4 passed the class (1A, 1B, 2C's) and 3 failed (1D and 2F).

Grades	A	B	C	D	F
All	9	12	8	4	11
Female Students	1	1	2	1	2

In labs, students were assessed based on occasional quizzes, in-class activities (weekly teamwork on problem solving), weekly labs to reinforce concepts seen in lecture, and comprehensive more complex lab assignments. Their participation and attendance were also counted in their overall assessment and final grade.

The following is a mapping of the outcomes of the course to the instruments used to assess the students, along with the observed % of success.

CS1301 OUTCOMES MAPPING

	Total	Exams	MT1	MT2	MT3	Final	Quizzes	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Level 1: knowledge and comprehension															
1.The history of computing															
2. The relation between computing and society, including main social, ethical, and legal issues															
3. Computing as a profession, from required knowledge and skills to major career options															
4. Computer representation of simple data types and operations, including operations with binary numbers															
5. Differences between programming languages															
6. Pseudocode of the use of multi-D arrays															
7. Pseudocode of the use of linked lists	ASSESSED	X	Q30: **85%**			Q11: **74.2%**									
Level 2: Application and Analysis															
1.To analyze problems and express solution algorithms in pseudocode, including a correct use of:	ASSESSED	X	X	X	X	X	X	X	X	X		X	X	X	X
Arithmetic and logical expressions	ASSESSED	X	Q6, 7, 8, 9, 10, 11, 12: **91.8%**	Q4: **83.5%**	Q1, 2, 3, 5: **83.4%**	Q1, 2, 3, 5: **90.8%**	X								
Simple I/O operations	ASSESSED	X		Q6: **43%**			X								
User-defined subprograms, including resursive methods	ASSESSED	X	Q25, 28, 29: **66.7%**	Q3, Q5: **88.5%**	Q1, 2, 3, 4, 7: **82.3%**	Q1, 2, 3, 4, 7: **93.3%**	X								
User-defined types	ASSESSED	X			Q4: **86.25%**	Q10, 11: **73.55%**	X								
2. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults	ASSESSED	X	Q6, 7, 8, 9, 13, 14, 15, 27: **87.5%**	Q2, 5, 6: **78%**	Q6, 7: **86.7%**	Q4, 5, 6, 7, 8, 9: **88.9%**	X								
3. Use teamwork roles and methods in the classroom															
Level 3 Outcomes: Synthesis and Evaluation															
1. Basic variable types such as integer, real number, character, string, 1-D array	ASSESSED	X	Q2, 3, 4, 17, 18, 19: **80%**	Q1, 2, 4, 6: **82.5%**	Q1, 2, 3, 6: **80.7%**	Q1, 2, 3, 4, 9: **91.3%**	X								
2. Assignment, arithmetic, and logical operations	ASSESSED	X	Q2, 5, 16: **75%**	Q4: **83.5%**	Q1, 2, 3, 5, 6: **84%**	Q1, 2, 3, 4, 5, 9: **90.2%**	X								
3. Basic control structures: if-then, for-loop, while-loop	ASSESSED	X	Q13, 14, 15, 20, 21, 22, 23, 24, 26: **62.2%**	Q2, 4: **86%**	Q1, 2, 3, 6: **80.7%**	Q1, 2, 3, 6, 9: **90.38%**	X								

We observe that Simple I/O operations were not assessed in a way that clearly showed students' performance on this topic. Often questions about simple I/O come up in quizzes or class questions that are not systematically assessed. Also, they may come up in other types of questions where the emphasis is on other outcomes such as loops, methods, etc. In lab though, we observed that students were performing very well in I/O topics.

Recommendation: add a series of questions in Midterm 1 about simple I/O to clear this topic.

		Discover Java	Logic game	Methods	Methods, Conditionals, Repetitions	File I/O	Testing and debugging	Arrays, File I/O, conditionals, repetitions	Board game and UDTs	Linked lists and sorting	Problem solving team work
	Labs	Minilab 1	Minilab 2	Minilab 3	Minilab 4	Minilab 5	Fixit Week	Comp 1	Comp 2	Comp 3	Weekly teamwork
Level 1: knowledge and comprehension											
1. Computer representation of simple data types and operations, including operations with binary numbers											
2. Technical aspects of computing, including memory, operating systems, editors, interpreters, compilers, debuggers, and virtual machine											
3. Differences among programming languages											
4. The purpose and use of exceptions											
5. Pseudocode and implementation in a programming language of the use of Multi-D arrays											
6. Pseudocode and implementation in a programming language of the use of Linked lists											
Level 2: Application and Analysis											
1. To analyze problems and express solution algorithms in pseudocode											
2.To implement pseudocode algorithms in a high-level language, including the correct use of:											
Arithmetic and logical expressions											
Simple I/O operations											
User-defined subprograms, including resursive methods											
User-defined types											
3. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults											
4. Use teamwork skills, including the use of teamwork roles											
Level 3 Outcomes: Synthesis and Evaluation											
1. Basic variable types such as integer, real number, character, string, 1-D array											
2. Assignment, arithmetic, and logical operations											
3. Basic control structures: if-then, for-loop, while-loop											
Overal performance		81.10%	99.70%	105.14%	92.40%	80.60%	82.30%	75.50%	92.30%	81.55%	Satisfactory

All outcomes were covered in classes and/or in labs. As shown in the above tables, all level-2 and level-3 outcomes were assessed, and some of the level-1's. The use of teamwork was daily and students were held accountable to their roles in teams. They were specifically assigned work to perform in teams once a week starting on week 6 in labs. The assessment of this part of the work was qualitative (see handouts of these activities for more information).

Regarding the performance of students in the above formal assessments of outcomes, here is a summary:

- Exams: All outcomes were attained by the end of the semester. Lower performances earlier in the semester show my testing of topics just covered: I regularly test my students on topics even just covered to give them an idea of what is expected of them, while reminding them that they are not yet expected to perform perfectly on these since we just started.
  At the final exam: the weak area was user-defined types. As commented earlier, this is not representative of the students ability in the area, as checked on non-graded in-class work, and as demonstrated by their good performance on the last two comprehensive labs of the semester on UDTs and linked lists.
- Labs: All outcomes were attained. The only weak point in performance was for comprehensive lab 1. This is typical: students are not used to this type of lab (longer lab, more complex, comprehensive) and tend to perform poorly the first time they have to complete one. Performance on the next two comprehensive labs was better.

4. Analysis by Instructor:

1. General comments:

The passing rate in the lab was lower than would normally be expected (we would have expected about 80% to be on par with regular semesters). The % of women is still low as well (ranging between 8% and 22%).

The performance on User Defined Types (UDTs) is marginal (86% at MT2 and 73% at the final exam). This is one of the last topic covered (before linked lists) and as a result, students did not get to practice taking exams on this topic. However, in non-graded classwork, they performed very well on this topic and were able to build UDTs without a problem. Recommendation: non-graded in-class work could become a graded assignment to provide an extra data point on students performance in CS1301 on this topic.

I find that some outcomes are buried in a category with multiple topics and prevent us from specifically diagnosing students' performance. For instance, 1D array manipulation is part of the same outcome as simpler types. These should be distinguished. Similarly, recursion is bundled with all about methods. Students may understand methods very well but struggle with recursion: we could split these to have a better picture of performance. Recommendation: I suggest revisiting the way we assess our outcomes, or maybe the way we lay them out so that we can have finer grained information.

2. Overall, the number of **passing students** was average, based on previous semesters.

3. We keep struggling with students' lack of timeliness in submitting work. I ignore "bumps in the road" if students show motivation and dedication moving forward during the semester, as I believe that things can happen.

4. In labs, this semester, we started new activities: we spent 3 weeks on introductions and ice-breakers (including coding ice-breakers). We used music in the lab. We started problem-solving Fridays in teams (mimicking interview questions). We had a week dedicated to debugging and testing (fix-it week). We intertwined weekly activities meant to reinforce concepts seen in class (or not for those not attending the lecture) with comprehensive labs that were more grounded in real-life applications and situations.

Recommendations:

1. TAs are crucial in the success of our students. Their skills are essential in delivering relevant content in labs and in providing relevant and timely guidance. It is important to make sure that the **TAs be properly prepared**: they should be knowledgeable in the programming languages taught in this course, but also familiar with the goal of the course and trained in pedagogy. Training that was put in place allowed us to have very good TAs and instructional teams in general, but we need to stress how important this training is so that it continues. Additionally, we recommend creating a pipeline of instructional assistants (IA, PL, TA) so that we ensure the continuity of quality of instruction semester after semester.
2. Facing a growing enrollment, in addition to larger lecture sections, we now also have larger lab sections. This is manageable, but I found it useful to have 2 TAs in a large lab (not just one), along with a PL and an IA.

Recommendations from previous reports and actions taken:

In the last report, the following was brought up:

1. Because of the change brought by the split into CS1301 and CS1101, the committee recommends studying the possible change in drop rate or trend: will students drop more? Less? Will students'

ability to drop only part of the "course" tell us more about the reasons for drops and possibly how to address them?

→ The attrition rate in these two sections of CS1301 and a lab is respectively 19%, 7%, and 6.4%, which shows that we managed to decrease the drop rate. However, 19% attrition in the 1st section (the largest and earliest one) is still not satisfactory and we need to work on understanding what drives students to drop. Recommendation: Often, we do not know why students drop because we do not see them again and they seldom answer our emails asking why they dropped. We should ask academic advisors to document why students drop when they ask for a signature to allow them to drop.

2. Outcomes should be revisited so that each course, CS1301 and CS1101, but standalone.

   → We did that and came up with two separate syllabi: they are very similar but some discrepancies exist, such as an emphasis on implementation vs. pseudo-coding, and the coverage of exceptions, which only happens in lab.

3. As many institutions reconsider the language of instruction, so should we.

   → We did consider the language of instruction and decided to stay with Java. However, our conversation led us to changing the language of instruction of CS3 from Java to Python, implemented starting in Summer 2018.

4. We should better assess teamwork.

   → In Fall 2018, I started formal and systematic weekly group work that was assessed. This gave the students more guidance about how to work in groups and it provided us a better understanding of how students were doing. Since it was a weekly activity (30 minutes every Friday with the same teams), we were able to provide consistent feedback and support. We still did not attach a grade to this work but were able to identify who needed help and provided such help.

CS 1301/1101 Introduction to Computer Science CQI Course Review - Fall 2018

Schuyler Manchester

1. Course Description

Students will learn to be active learners, understand the motivations for computing, basic concepts of algorithms, basic computer organization, and impacts of computing. They will develop problem-solving skills, implement solutions to computing problems in a high-level programming language, and build team skills, critical-thinking skills, and professionalism.

Knowledge and Abilities Required Before Entering the Course: Students entering the course are not required to have a background in Computer Science or programming. They should be familiar with topics from Pre-calculus, including algebraic functions, proofs, and base representations of numbers.

Prerequisite: MATH 1508 or MATH 1411 with a grade of C or better.

Textbook: Programming in Java, by Zybooks, available at zybooks.zyante.com

2. Learning Outcomes

Level 1: Knowledge and Comprehension

Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to describe, at a high level:

1.The history of computing
2. The relation between computing and society, including social, ethical, and legal issues
3. Computing as a profession, from required knowledge and skills to major career options
5. Computer representation of simple data types and operations, including operations with binary numbers
6. Differences among programming languages
7. Pseudocode of the use of Multi-D arrays
8. Pseudocode of the use of Linked lists

Level 2: Application and Analysis

Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able:
1.To analyze problems and express solution algorithms in pseudocode, including a correct use of:

a. Arithmetic and logical expressions
b. Simple I/O operations
c. User-defined subprograms, including recursive methods
d. User-defined types

2. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults
3. Use teamwork roles and methods in the classroom

Level 3 Outcomes: Synthesis and Evaluation

Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course, students will be able to use the syntax and semantics of a higher-level language to express solutions to programming problems, including the pseudocode
correct use of:
1. Basic variable types such as integer, real number, character, string, 1-D array
2. Assignment, arithmetic, and logical operations
3. Basic control structures: if-then, for-loop, while-loop

3. Observations and Instruments

CS 1301

There were 38 students who registered for the class at the time of the first day of classes.

- 8 student dropped
- Out of the 30 remaining
    - 23 passed the class (77%): 11 As, 9 Bs, 3 Cs
    - 7 did not pass the class (23%): 4 Ds and 3 Fs
    - 9 were female students (30%) and 8 of them passed the class (88%): 2 As, 4 Bs, 2 Cs, 1 F.

Students were evaluated throughout the semester via quizzes (all online), three midterm exams, and a comprehensive final exam. Students were also evaluated based on their attendance and participation in class and in lab.

Learning Outcome	Total	Extra Assignment	Exams	E1	E2	E3	Final	Quizzes	GQ1	GQ2	GQ3	GQ4
Level 1: Knowledge and Comprehension												
1.The history of computing	assessed	X										
2. The relation between computing and society, including social, ethical, and legal issues	assessed	X										
3. Computing as a profession, from required knowledge and skills to major career options	assessed	X										
5. Computer representation of simple data types and operations, including operations with binary numbers	assessed			Q2 - Q11			Q4, Q12, Q28		X			
6. Differences among programming languages	assessed	X										
7. Pseudocode of the use of Multi-D arrays	assessed			Q16-21	Q2, Q10, Q13, Q16	Q7, Q11	Q15, Q17				X	
8. Pseudocode of the use of Linked lists	assessed					Q2, Q4	Q21					X
Level 2: Application and Analysis												
1.To analyze problems and express solution algorithms in pseudocode, including a correct use of:	assessed			Q1								
a. Arithmetic and logical expressions	assessed			Q6-Q9	Q13, Q15		Q5-Q7, Q9		X	X		
b. Simple I/O operations	assessed			Q22	Q12, Q15		Q14			X	X	
c. User-defined subprograms, including recursive methods	assessed				Q3, Q4, Q9, Q11, Q13, Q14	Q5-Q7	Q1-Q2, Q20, Q22-Q25				X	X
d. User-defined types	assessed				Q1, Q5-Q8	Q2, Q4,	Q3, Q8,					X

						Q10-Q12	Q11, Q18, Q19, Q26, Q27					
2. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults	assessed					Q1, Q3, Q8, Q9	Q29					
3. Use teamwork roles and methods in the classroom	assessed	X										
Level 3 Outcomes: Synthesis and Evaluation												
1. Basic variable types such as integer, real number, character, string, 1-D array	assessed			Q2 - Q11	Q2, Q10, Q13, Q16		Q9, Q13, Q15, Q16		X	X	X	X
2. Assignment, arithmetic, and logical operations	assessed			Q6-Q9		Q6			X	X	X	X
3. Basic control structures: if-then, for-loop, while-loop	assessed			Q12-Q15	Q12	Q6	Q10		X	X	X	X

All outcomes were covered in classes and/or in labs. The use of teamwork occurred primarily throughout the lab (see separate report).

Regarding the performance of students in the above formals assessments of outcomes, here is a summary:

- **Exams:** Most outcomes were attained. We focus on the final assessment of these (at the final exam) since it was comprehensive and I emphasized the flexibility in learning speed to show my focus on grading and assessing the acquisition of skills rather than the speed at which students were acquiring these.
- **At the final exam:** the weak areas were:
    - (1) writing code to combine two arrays (55%), which in hindsight might have been due to the uniqueness of this question compared to other array questions. In Q20 (78%) and Q22 (74%) we saw substantial improvement and tested multiple components of understanding of arrays
    - (2) Values of variables with multiple assignment and post-fix ++ operations (43%): My understanding as to why this was so challenging for the final is because we went over this in detail early on in the semester, and although we remained using the ++ operator, it was almost exclusively used in for loops.

CS 1101

There were 46 students who registered for the class at the time of the first day of classes.

- 6 of the students never showed up;
- Out of the 40 remaining
    - 34 passed the class (85%): 15 As, 11 Bs, and 8 Cs
    - 6 did not pass the class (15%): 6 Fs
    - 10 were female students (25%) and 8 of them passed the class (80%): 6 As, 2 Bs, and 2 Fs.

Students were evaluated throughout the semester via across lab assignments, comprehensive lab assignments, in lab activities, and assigned homework readings. Students were also evaluated based on their attendance and participation within the lab.

Learning Outcome	Total	Extra Assignment	Lab 2 - user input & boolean logic	Lab 3 - switch statements & loops	Comprehensive Lab 1	Lab 4 - arrays	Lab 5 - methods	Comprehensive Lab 2	Lab 6 - objects & file reading	Lab 7 - objects & recursion	Lab 8 - linked lists	Comprehensive Lab 3
Level 1: Knowledge and Comprehension												
1. Computer representation of simple data types and operations, including operations with binary numbers	assessed		X									
2. Technical aspects of computing, including memory, operating systems, editors, interpreters, compilers, debuggers, and virtual machine	assessed	X										
3. Differences among programming languages	assessed	X										
4. The purpose and use of exceptions	assessed	X								X		
5. Pseudocode and implementation in a programming language of the use of Multi-D arrays	assessed					X		X				
6. Pseudocode and implementation in a programming language of the use of Linked lists	assessed										X	
Level 2: Application and												

Analysis												
1.To analyze problems and express solution algorithms in pseudocode	assessed							X				
2. To implement pseudocode algorithms in a high-level language, including the correct use of:	assessed							X				
a. Arithmetic and logical expressions	assessed		X			X						
b. Simple I/O operations	assessed		X	X		X		X	X	X		
c. User-defined subprograms, including recursive methods	assessed						X	X	X	X	X	X
d. User-defined types	assessed								X	X	X	X
3. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults	assessed	X								X		
4. Development of teamwork skills, including the use of teamwork roles.	assessed	X										
Level 3 Outcomes: Synthesis and Evaluation												
1. Basic variable types such as integer, real number, character, string, 1-D array	assessed		X	X	X	X	X	X	X	X	X	X
2. Assignment, arithmetic, and logical operations	assessed		X	X	X	X	X	X	X	X	X	X
3. Basic control structures: if-then, for-loop, while-loop	assessed		X	X	X	X	X	X	X	X	X	X

All outcomes were covered in classes and/or in labs.

Regarding the performance of students in the above formals assessments of outcomes, here is a summary:

- **Labs:**
    - **Lab 6 objects & file reading** proved to be one of the most challenging labs for students (63% average). This lines up with the fact that a number of students still didn't have a good grasp on the concept of classes & objects, and this lab was their first assessment with this concept. Further labs, although they had more challenging components (recursion and linked lists) had a high score as more and more students got a better grasp of objects.
- **Comprehensive Labs:**
    - **Comprehensive Lab 3** had a substantial average drop (78%) compared to the first two (90% and 86% respectively). This is largely due to a number of students not completing the assignment at all, 5 of the zeroes were students that failed the class (likely realizing they would have to retake the course).

4. Analysis by Instructor:

1. Overall the number of **passing students** was average based on previous semesters.
2. Using a lightweight / no overhead IDE (repl.it) proved to be a very valuable tool to use in class. It made it very easy to share code snippets with the students, and also provide a template starting point for everyone. Additionally, it was valuable that students didn't have to worry about setting up a Java environment and could work on any machine and pick up where they left off. This allowed students to focus on learning new concepts instead of constantly battling Java environment issues (they still learned this skill set in the lab).
3. The online textbook was a very helpful tool, it reduced overhead of grading, and was simple enough for students to use.

5. Recommendations

1. It would be very advantageous to students if we could guarantee they had a matching lecture to lab, so we can ensure continuity of education.
2. The value and impact TAs and IAs have on students is fairly large. Giving them some formal training on speaking to large classes (ensuring to broadcast voice, move about the room, etc.) and general teaching paradigms could be very helpful.

I. COURSE DESCRIPTIONS

CS 1301 Introduction to Computer Science - Fall 2018

Course Objectives: Students will learn to be active learners, understand the motivations for computing, basic concepts of algorithms, basic computer organization, and impacts of computing.
They will develop problem-solving skills, implement solutions to computing problems in a high level programming language, and build team skills, critical-thinking skills, and professionalism. **Prerequisite:** MATH 1508 or MATH 1411 with a grade of C or better.

Knowledge and Abilities Required Before Entering the Course: Students entering the course are not required to have a background in Computer Science or programming. They should be familiar with topics from Pre-calculus, including algebraic functions, proofs, and base representations of numbers.

Logistics

Lecture sessions: Tuesdays and Thursdays, 10:30- 11:50am at CCSB 1.0704.
Instructor: Natalia Villanueva Rosales, e-mail: nvillanuevarosales@utep.edu, office: CCSB Room 3.0508, phone: (915) 747-8643.
Office hours: Tuesdays and Thursdays, 9:30-10:30am and 1:30-2:30pm and by appointment outside this time. Please use email to contact instructor.
Teaching Assistant: Adriana Camacho, accamacho@miners.utep.edu
TA Office hours: MW 10:30am-12:00pm, Tu 2:00pm-3:00pm
Textbook: Programming in Java, by Zybooks.
Available at zybooks.zyante.com. To subscribe to your textbook, please enter the following code: **UTEPCS1301VillanuevaRosalesFall2018**
Online platform: Blackboard
Software: Software used in this course is available on the desktop computers in the main computer lab and in the two instructional labs on the first floor. To use the course software on your home or laptop computer, instructions will be given in the labs and available online on our piazza page.

Learning Outcomes

Level 1: Knowledge and Comprehension. Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to describe, at a high level:

1. The history of computing
2. The relation between computing and society, including social, ethical, and legal issues
3. Computing as a profession, from required knowledge and skills to major career options

4. Computer representation of simple data types and operations, including operations with binary numbers
5. Differences among programming languages
6. Pseudocode of the use of Multi-D arrays
7. Pseudocode of the use of Linked lists

Level 2: Application and Analysis. Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able:

1. To analyze problems and express solution algorithms in pseudocode, including a correct use of:
    a. Arithmetic and logical expressions
    b. Simple I/O operations
    c. User-defined subprograms, including recursive methods
    d. User-defined types
2. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults
3. Use teamwork roles and methods in the classroom

Level 3 Outcomes: Synthesis and Evaluation. Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course, students will be able to use the syntax and semantics of a higher-level language to express solutions to programming problems, including the pseudocode correct use of:

1. Basic variable types such as integer, real number, character, string, 1-D array
2. Assignment, arithmetic, and logical operations
3. Basic control structures: if-then, for-loop, while-loop

Grading

Grades are communicated to students in a timely manner. It is the students' responsibility to keep track of their grades by compiling the grades they receive. Your semester grade will be based on a combination of homework assignments, quizzes, class participation, 3 mid-term exams, CS engagement, and a final exam.

The **approximate** percentages are as follows:

- 20% Homework
- 15% Quizzes
- 50% Exams (3 mid-term exams and 1 final exam)
- 5% Student Engagement in Computer Science

- 10% Class participation (includes on-time lecture attendance, active participation in class, completion of any quizzes for attendance and survey purposes)

The nominal percentage-score-to-letter-grade conversion for CS 1301 is as follows:

- 90% or higher is an A
- 80-89% is a B
- 70-79% is a C
- 60-69% is a D
- Below 60% is an F

NOTE: You must earn a C or better in each of these two courses, CS1301 and CS1101, to continue to the next course in this sequence, which is CS2401.

Expectations

Class Participation: Attendance and participation in all lecture sessions are critical factors of your success in this course. Students should be **on time** for all scheduled sessions and **attend the entire session**. Attendance will be taken at every session and will count towards your class participation grade. Students should **notify the instructor prior to missing a session** if at all possible, and certainly right after if earlier was not possible. The instructor will allow **two unexcused absences per semester** before having the option to deduct points from the final grade. It is the student's responsibility to obtain the content covered during missed class(es). Participation points also include completing post-lecture and post-labs online quizzes (when requested) that are administered as surveys to monitor students' overall progress and potential struggles.

Quizzes: The purpose of each **quiz** is to ensure that you are staying current with the weekly reading assignments and video lectures and to verify that you have acquired the skills developed in class. Quizzes will usually be on-line. There will be **no makeup** on non-justified missed quizzes.

Homework: Reading and homework assignments will be announced in class and/or posted on Blackboard. If you miss a lecture session, it is your responsibility to find out what you missed. You should expect to spend **at least four hours per week outside of lecture** on reading and homework. Most of your homework will be work assigned on your online Zybook: completing the assigned activities on time will be crucial to your success in the class (since these activities prepare you for classwork) and to getting a good grade (since late completion will be penalized).

Exams: There will be 3 midterm exams and one final exam. All four exams together will weigh 50% of your overall final grade for CS1301. Because the exams contribute so heavily to your total grade, it is vital that you do well on them. If you have test-taking difficulties in general, or if you have difficulties with our tests in particular, please let me know as soon as possible and/or request appropriate accommodation from UTEP's

Center for Accommodation and Students' Services. The purpose of the **midterm exams** is to allow you to demonstrate mastery of course concepts covered thus far during the semester. Mid-term exams will take place during the regular lecture session and are tentatively scheduled to be held around week 6, week 10, and week 14. Make-up exams will be given only in extremely unusual circumstances. If you must miss an exam, please meet with an instructor, **before the exam**. The **final exam** will be comprehensive. You must score 65% or better on the final exam to pass this course. You must take the final exam during the time shown in the schedule for the lecture section that you normally attend. Do not "drop in" to another section: there will not be a copy of the exam for you. This is University policy. If you have a scheduling conflict (e.g., if you are taking a final at EPCC) or if you are scheduled for three final exams in one day, see your instructor in advance for accommodation.

Student Engagement in Computer Science: During the course of the semester, you must engage as a computer scientist in activities as shown below, in a way that you cumulate at least 7 points (towards your final grade).

Possible activities (along with the number of points each yields) include (but are not limited to):

2 points for each of the following:

- Send a video to Dr. Villanueva Rosales about a CS topic of your preference and how it impacts our everyday life to be presented to the class;
- Write a summary of a seminar you attended (proof of attendance needs to be provided as well);
- Attend 6 workshops provided by undergraduate TAs;
- Participate in a research project as a research human subject;
- Participate in a Department's outreach as a volunteer student;
- Design a video or a presentation about a specific career in Computer Science;

3 points for each of the following:

- Be an active participant in Google IgniteCS program (or equivalent);
- Be an active undergraduate researcher in one of the Computer Science Research labs and present to the class your work.

Resources

Special Accommodations: If you have a disability and need classroom accommodations, please contact the Center for Accommodations and Support Services (CASS) at 747-5148 or by email to cass@utep.edu, or visit their office located in UTEP Union East, Room 106. For additional information, please visit the CASS website at www.sa.utep.edu/cass. CASS' staff are the only individuals who can validate and if need be, authorize accommodations for students with disabilities.

Scholastic Dishonesty: Any student who commits an act of scholastic dishonesty is subject to discipline. Scholastic dishonesty includes, but not limited to cheating, plagiarism, collusion, submission for credit of any work or materials that are attributable to another person. **Cheating** is copying from the test paper of another student. Communicating with another student during a test to be taken individually. Giving or seeking aid from another student during a test to be taken individually. Possession and/or use of unauthorized materials during tests (i.e. crib notes, class notes, books, etc.). Substituting for another person to take a test. Falsifying research data, reports, academic work offered for credit. **Plagiarism** is using someone's work in your assignments without the proper citations. Submitting the same paper or assignment from a different course, without direct permission of instructors. To avoid plagiarism, see: http://sa.utep.edu/osccr/wp-content/uploads/sites/8/2012/09/AvoidingPlagiarism.pdf. **Collusion** is unauthorized collaboration with another person in preparing academic assignments

NOTE: When in doubt on any of the above, please contact your instructor to check if you are following authorized procedure.

CS 1101 Introduction to Computer Science Lab - Fall 2018

Course Objectives: Students will learn the foundations of algorithmic thinking and algorithm development, and learn how to implement them in a variety of languages. They will also learn to be active learners. They will develop problem-solving skills and build team skills, critical thinking skills, and professionalism

Prerequisite: MATH 1508 or MATH 1411 with a grade of C or better.

Knowledge and Abilities Required before Entering the Course: Students entering the course are not required to have a background in Computer Science or programming. They should be familiar with topics from Pre-calculus, including algebraic functions, proofs, and base representations of numbers.

Logistics

Lab sessions: Tuesdays and Thursdays, 3:00- 4:20pm CCSB 1.0704.

Instructor: Natalia Villanueva Rosales, e-mail: nvillanuevarosales@utep.edu, office: CCSB Room 3.0508, phone: (915) 747-8643.

Office hours: Tuesdays and Thursdays, 9:30-10:30am and 1:30-2:30pm and by appointment outside this time.

Teaching Assistant: Adriana Camacho, accamacho@miners.utep.edu

TA Office hours: MW 10:30am-12:00pm, Tu 2:00pm-3:00pm.

NOTE: Lab sessions were led by TA with assistance of Instructional Assistant and one peer-leader.

Textbook: Programming in Java, by Zybooks.
Available at zybooks.zyante.com. To subscribe to your textbook, please enter the following code: **UTEPCS1301VillanuevaRosalesFall2018**

Online platform: Blackboard

Software: Software used in this course is available on the desktop computers in the main computer lab and in the two instructional labs on the first floor. To use the course software on your home or laptop computer, instructions will be given in the labs and available online on our piazza page.

Important lab rule about using your personal laptop computers: You are allowed to use your personal computer instead of UTEP's desktop to complete the labs assigned to you. However it is essential that you be able to **demo** your work anytime we ask you for it in lab. We will not accept that "your work is on your laptop and you cannot produce it at the time we request it". To avoid such situation you could for instance use Dropbox and hence make sure that you can access your work from anywhere.

Learning Outcomes

Level 1: Knowledge and Comprehension. Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply

basic definitions. On successful completion of this course, students will be able to describe, at a high level:

2. Computer representation of simple data types and operations, including operations with binary numbers
3. Technical aspects of computing, including memory, operating systems, editors, interpreters, compilers, debuggers, and virtual machine
4. Differences among programming languages
5. The purpose and use of exceptions
6. Pseudocode and implementation in a programming language of the use of Multi-D arrays
7. Pseudocode and implementation in a programming language of the use of Linked lists

Level 2: Application and Analysis. Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able:

1. To analyze problems and express solution algorithms in pseudocode.
2. To implement pseudocode algorithms in a high-level language, including the correct use of:
    e. Arithmetic and logical expressions
    f. Simple I/O operations
    g. User-defined subprograms, including recursive methods
    h. User-defined types
4. To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults
5. Use teamwork roles and methods in the classroom

Level 3 Outcomes: Synthesis and Evaluation. Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course students will be able to use the syntax and semantics of a higher-level language to express solutions to programming problems, including the correct use of:

4. Basic variable types such as integer, real number, character, string, 1-D array
5. Assignment, arithmetic, and logical operations
6. Basic control structures: if-then, for-loop, while-loop

Grading

Grades are communicated to students in a timely manner. It is the students' responsibility to keep track of their grades by compiling the grades they receive. Your semester grade will be based on a combination of lab assignments, lab homework, quizzes, class participation.

The approximate percentages are as follows:

- 70% Lab assignments and homework (including demos and comprehensive labs)
- 15% Zybook exercises
- 15% Lab participation (includes on-time attendance, active participation in lab discussion, activities including teamwork, and completion of any quizzes for attendance and survey purposes)

The nominal percentage-score-to-letter-grade conversion for CS 1301 is as follows:

- 90% or higher is an A
- 80-89% is a B
- 70-79% is a C
- 60-69% is a D
- Below 60% is an F

NOTE: You must earn a C or better in each of these two courses, CS1301 and CS1101, to continue to the next course in this sequence, which is CS2401.

Expectations

Lab Assignments: Lab assignments are designed to allow you to practice the topics that constitute the outcomes of this course. Assignments will be a mix of:

- Problems to be solved without computers to practice problem solving and algorithm design;
- Programming assignments.

Deadlines for lab assignments (short and comprehensive) will be clearly specified in the description of each assignment. Assignments turned in up to three days late will have scores reduced by 10% for each day of lateness. When assessing labs, TAs and Instructional Assistants will spend 5 to 10 minutes with each students asking demos of the functionality of your program as well as topics covered in the assignments. Questions will be asked regardless of whether you completed the assignment or not. This allows you flexibility, in case something happened and you were not able to complete an assignment, to make up for some points.

Lab Participation: Attendance and participation in all lecture sessions are critical factors of your success in this course. Students should be **on time** for all scheduled sessions and **attend the entire session**. Attendance will be taken at every session and will count towards your class participation grade. Students should **notify the instructor prior to missing a session** if at all possible, and certainly right after if earlier was not possible. The instructor will allow **two unexcused absences per semester** before having the option to deduct points from the final grade. It is the student's responsibility to obtain the content covered during missed class(es). Participation points also include

completing post-lecture and post-labs online quizzes (when requested) that are administered as surveys to monitor students' overall progress and potential struggles.

II. COURSE APPROACH

Authentic and contextualized problem–based learning.

Topics introduced in the class were motivated and exemplified using real-world scenarios related to the community. For example, a big challenge in campus is parking. Every semester students struggle to find parking. Students were encouraged to envision solutions where computers and algorithms would improve parking conditions at UTEP, brainstorm and make a group presentation at the beginning of the semester. Many of their approaches, e.g., use of sensors in parking spots, were used later in the semester to introduce loops and conditions under self-parking scenarios. Exams and assignments made use of these scenarios, already familiar to students.

Hands-on activities

For every new topic introduce, students were given hands-on activities to graphically and physically (when possible) demonstrate the intuition behind the operation or technique being explained. Activities most of the time included teamwork and the opportunity to students to also create their own examples and share with the group. Small groups were preferred where students would help each other and discuss their findings. Teamwork was always monitored by the instruction team.

Instruction team

We were fortunate to have a teaching assistance (TA), instructional assistant (IA; an UG student), and a peer leader. Having this support was instrumental to ensure students always had assistance both in the classroom, lab and outside instruction time. In conjunction with other CS1 sections, students had access to TAs from 9am-5pm almost every day of the week. Due to the change of having only one lab for every lecture (previously two), students were encouraged and required to demo their comprehensive labs during office hours of the TA or the IA. This change also promoted attendance during office hours. Students struggling in the course were personally invited by the instruction team to come to office hours. Individual attention seemed to work better in this case. In addition, students missing a class, and/or the lecture received an email the day of the session or the following day indicating that we were aware of their absence and the topics covered. Students were also emailed when they had more than two absences not justified. The instruction team met at least once a week, meetings with TA were done twice a week to ensure the contents of the lab were synchronized with the contents in the lecture. The instruction team also attended lectures.

In addition, a lead TA funded by the NSF-funded RED grant did a weekly training to all the TAs and observations during labs. These activities were instrumental to enable instructors to standardize the outcomes for the course across the three sections and corresponding labs.

Semi-automated grading

The use of Blackboard was key to create quizzes and exams online (when possible). By using questions where they had to complete code, answer questions that required doing tracing or analysis on some algorithms, students were able to get immediate feedback. In addition, students used exams from previous years as practice before the exam. Once students got familiar with Blackboard, reusing materials for practice and doing exams online were not an issue. Savings in grading time was allocated to generate materials and exercises for individual practice.

III. COURSE DEMOGRAPHICS AND FINAL GRADES

III.1. CS 1301

Total students 41 by the end of the semester – 6 females (14.6%).

Initially, 47 students were registered the first day of classes. Five students dropped during the first two weeks of the semester (including one female student) including three that never attended lectures. T**wo students dropped** during the second half of the class.

A	15
B	12
C	9
D	4
F	1
Average	83.5

III.2. CS 1101

Total students by the end of the semester 52 – 5 females (9.61%).

The first day of class, 55 students were enrolled in the class. From those, three students dropped, including one student taking the lab in our section and the lecture with a different instructor.

A	15
B	11
C	11
D	4
F	4
Average	72.47

One of the reasons frequently listed for not being able to dedicate enough time to the class outside the sessions was over commitment with activities outside school (e.g., family or work responsabilities).

IV. COURSE OUTCOMES COVERAGE

CS 1301		CS 1101		
Outcome#	Short description	Outcome#	Short Description	Avg.
Level 1:	Knowledge and Comprehension - Students will be able to describe, at a high level:			
1.1.	The history of computing			97.2
1.2.	The relation between computing and society, including social, ethical, and legal issues			98.35
1.3.	Computing as a profession, from required knowledge and skills to major career options			87.1
1.4.	Computer representation of simple data types and operations, including operations with binary numbers	1.1	Computer representation of simple data types and operations, including operations with binary numbers	91.69
1.5.	Differences among programming languages	1.3	Differences among programming languages	78.9
1.6.	Pseudocode of the use of Multi-D arrays	1.5	Pseudocode of the use of Multi-D arrays	83.85
1.7.	Pseudocode of the use of Linked lists	1.6	Pseudocode and implementation in a programming language of the use of Linked lists	90.38
		1.2	Technical aspects of computing, including memory, operating systems, editors, interpreters, compilers, debuggers, and virtual machine	96.09
		1.4	The purpose and use of exceptions	79.33

Level 2:	Application and Analysis. Students will be able to:			
2.1.	Analyze problems and express solution algorithms in pseudocode, including a correct use of:	2.1.	Analyze problems and express solution algorithms in pseudocode, including a correct use of:	87.6
2.1.a.	Arithmetic and logical expressions	(Imp.) 2.2.a	Arithmetic and logical expressions	83.35
2.1.b.	Simple I/O operations	2.2.b	Simple I/O operations	90.51
2.1.c.	User-defined subprograms, including recursive methods	2.2.c	User-defined subprograms, including recursive methods	71.73
2.1.d.	User-defined types	2.2.d	User-defined types	83.35
2.2.	To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults	2.3	To use testing and debugging strategies, including black-box and white-box testing, test drivers, stubs and test suites, to identify software faults	81.09
2.3.	Use teamwork roles and methods in the classroom	2.4	Use teamwork roles and methods in the classroom	97
Level 3:	Synthesis and Evaluation. Students will be able to use the syntax and semantics of a higher-level language to express solutions to programming problems, including the pseudocode correct use of:			
3.1.	Basic variable types such as integer, real number, character, string, 1-D array	3.1.	Basic variable types such as integer, real number, character, string, 1-D array	88.45
3.2	Assignment, arithmetic, and logical operations	3.2	Assignment, arithmetic, and logical operations	83.7
3.3.	Basic control structures: if-then, for-loop, while-loop	3.3.	Basic control structures: if-then, for-loop, while-loop	81.76
	Average			86.92

Table 1. Course outcomes for CS1301 and CS1101 and coverage according to the average in students presenting exam and labs.

In average, all the outcomes were covered by evaluations and lab submissions either in the lecture or the lab sessions. Mostly in both of them – lectures introduced the topic and lab sessions provided opportunities for implementation and hands-on activities. In

average the outcomes were covered in 87%. The outcomes 1.1 (history of computing), 1.2 (computing and society) and 2.3 (teamwork) obtained higher averages, with over 97% averages. These topics are at level 1 and 2 and covered at the beginning of the course when most of the students were submitting their homework and engaged. On the other hand, the outcomes with less average in evaluations- between 70% and 80% were: 1.4 in the lab (exceptions), 1.5 (differences among programming languages) and 2.1.c. (recursion). These outcomes were covered towards the lend of the semester (last three month). While some of the topics (i.e. recursion) may be considered as hard to grasp, my observations are that lower grades are affected by attendance and the time students dedicate to the class. In fact, for the last three labs, the average was calculated based on students submitting the labs. Lab submission increased from less than 10% to over 60% in the last three weeks. In general, all the outcomes were successfully covered.

V. OBSERVATIONS AND RECOMMENDATIONS

Coverage of topics early in the semester.

Based on the results of outcomes coverage considering grades and attendance, it is recommended to cover most of the topics early in the semester and before the last three weeks of the semester. Time is provided throughout the semester to reviews and more practice that could be left for the last three weeks, with more independent work on topics that need reinforcement.

Instruction team.

Having IAs and peer-leaders in addition to TAs was instrumental in providing better assistance to students. The IA assigned to the course was in charge of assisting students during the lab (a large lab with more than 50 students) and also reaching out to students missing classes or with low grades. Personal attention made a difference to those students.

Academic dishonesty

A case of academic dishonesty involving plagiarism of code found online was found and reported to the Office of Student Conduct and Conflict Resolution and was handled properly by TA and promptly by the office. This incident happened towards the last half of the course. Lessons learned from this is that students should be provided with specific examples and reminders throughout the course as per the meaning of plagiarism and collusion. The problem found online was one not applied in the context of UTEP as other problems and it was a generic one that could be found. It is recommended to use new problems that cannot be found online.

Lab and Lecture with same instructor

The lecture and labs of CS1 have several sections. Students attending both lecture and lab with the same instructor are effectively covering the same topics in lectures and labs sessions. Students in different sections may see topics at different times which may be a disadvantage to some students. Students in this situation have communicated this

concern verbally and it is recommended to make efforts to enroll students in the two courses under the same instructor.

Teamwork

Having students in teams to perform coding (during the lab session) tasks seemed engaging, beneficial and helpful. Even though these activities were not graded, it helped solidify concepts and have students who did understand help those who might have not understood.

Continuous evaluation

In addition to exams, homework and lab assignments, adding a weekly survey and quiz for the lab (as it was done for the class/lecture) was beneficial to have a sense of the understanding of the topic being introduce before moving into the next one. It is recommended to have continuous feedback from students, which can be done in Blackboard to address the increasingly large groups observed in CS1.

CS 2401 Elementary Data Structures and Algorithms
Course Review
Fall 2018
Mahmud Shahriar Hossain

Course Description

CS 2401 focuses on basic data structures and fundamental algorithms. The course is the second fundamental course for students majoring in Computer Science. Students learn about basic algorithms, including searching and sorting; elementary abstract data types including linked lists, stacks, queues and trees; and basic algorithm analysis. A significant part of the course contains practical laboratory sessions. Students complete a series of programming assignments using Java programming language to practice and enhance their understanding about the concepts taught in the lecture sessions. Laboratory assignments are designed to develop skills in problem solving, implementation, and debugging.

Prerequisite: CS 1301 and CS 1101, or equivalents of CS 1301 and CS 1101. Students must secure C or better in the prerequisites to continue with CS 2401.

Text

Introduction to Java Programming, 10th edition, Comprehensive Version, by Y. Daniel Liang.

Learning Outcomes

Level 3: Synthesis and Evaluation:

1. Design and implement solutions to computational problems using the following data structures:
    a. multi-dimensional arrays;
    b. lists implemented as arrays or linked lists;
    c. stacks;
    d. queues;
    e. binary trees and binary search trees.

Level 2: Application and Analysis:

2. Describe, implement, and use the following concepts:
    a. classes, subclasses, and inheritance
    b. encapsulation and information hiding
3. Describe, implement, and use the following algorithms:
    a. sequential and binary search
    b. quadratic and O(n log n) sorting
    c. string manipulation and parsing
4. Describe and trace computer representation and memory allocation of:
    a. integers, real numbers, arrays and objects
    b. methods, including recursive methods and the use of activation records
5. Use basic notions of algorithm complexity:
    a. use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm

b. determine the best, average and worst-case behaviors of a simple algorithm
c. assess time and space trade-offs in algorithms

6. Use recursion and iteration as problem solving techniques

Instruments

The primary instruments of evaluating student performance in the course were three midterm exams, a comprehensive final exam, and nine laboratory assignments. A small portion of the grades was based on in-class quizzes and lab attendance. However, quizzes and attendance are not used as instruments to evaluate the course.

Median score values (converted to percentage) are reported in the following table to assess the outcomes. The term "Med:" in the table refers to "median score".

Outcome	Level	Test 1	Test 2	Test 3	Other	Final Exam	Result
3.1a Design and implement solutions to computational problems using the following data structure: multi-dimensional arrays;	3	Q5: Med: 70%			Lab 1 (1D) Med: 100% Lab 2 Med: 85% Lab 3 Med: 84%	Q1 Med: 100% Q3 Med: 100%	Satisfactory
3.1b Design and implement solutions to computational problems using the following data structure: list implementation of arrays or linked lists;	3	Q4 Med: 30% Q6 Med: 70% Q7 Med: 80%	Q2 Med: 100%		Lab 4 Med: 90% Lab 5 Med: 76% Lab 6 Med: 83%		Satisfactory
3.1c Design and implement solutions to computational problems using the following data structure: stacks;	3			Q1 Med: 67% Q2 Med: 100%	Lab 8 Med: 87%	Q16 Med: 100% Q20 Med: 100%	Satisfactory
3.1d Design and implement solutions to computational problems using the following data structure: queues;	3			Q1 Med: 67%	Lab 9 Med: 88%	Q20 Med: 100%	Satisfactory
3.1e Design and implement solutions to computational problems using the following data structure: binary trees and binary search trees;	3			Q3: Med: 33% Q4: Med: 100% Q5: Med: 80%		Q5 Med: 100% Q8 Med: 100%	Satisfactory
2.2a Describe, implement, and use the following concepts: classes, subclasses, and inheritance	2	Q2 Med: 50% Q7 Med: 80%			Lab 3 Med: 84% Lab 4 Med: 90%		Satisfactory
2.2b Describe, implement, and use the following concepts: encapsulation and information hiding	2				Lab 4 Med: 90% Lab 5 Med: 76%	Q9 Med: 100%	Satisfactory
2.3a Describe, implement, and use the following algorithms: sequential and binary search	2	Q1 Med: 70%			Lab 7 Med: 84%	Q5 Med: 100% Q6 Med: 100% Q7 Med: 100%	Satisfactory

Outcome	Level	Test 1	Test 2	Test 3	Other	Final Exam	Result
2.3b Describe, implement, and use the following algorithms: quadratic and O(n log n) sorting	2		Q7 Med: 100%			Q11 Med: 100% Q12 Med: 100% Q13 Med: 100% Q14 Med: 100%	Satisfactory
2.3c Describe, implement, and use the following algorithms: string manipulation and parsing	2		Q6 Med: 90%		Lab 6 Med: 83%	Q17 Med: 100%	Satisfactory
2.4a Describe and trace computer representation and memory allocation of: integers, real numbers, arrays and objects	2	Q2 Med: 50%			Lab 2 Med: 85% Lab 3 Med: 84% Lab 4 Med: 90%		Satisfactory
2.4b Describe and trace computer representation and memory allocation of: methods, including recursive methods and the use of activation records	2	Q3 Med: 66%	Q2 Med: 100% Q3 Med: 100% Q6 Med: 90%	Q3 Med: 33%	Lab 6 Med: 83%	Q2 Med: 100% Q3: Med: 100% Q6: Med: 100%	Satisfactory
2.5a Use basic notions of algorithm complexity: use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm	2		Q1 Med: 60% Q4 Med: 33% Q5 Med: 100%		Lab 7 Med: 84%	Q7 Med: 100% Q12 Med: 100% Q13: Med: 100% Q14: Med: 100%	Satisfactory
2.5b Use basic notions of algorithm complexity: determine the best, average and worst-case behaviors of a simple algorithm	2		Q1 Med: 60% Q4 Med: 33% Q5 Med: 100%		Lab 7 Med: 84%		Satisfactory
2.5c Use basic notions of algorithm complexity: assess time and space trade-offs in algorithms	2					Q15: Med: 0% (Average: 46%)	Not satisfactory. More instruments ar required.

Analysis by Instructor:

The table above presents the results of mapping educational outcomes to assessment. My observations and recommendations are as follows.

Observations:

1. All outcomes were satisfied, except for outcome 2.5c. The median of 0% in the only instrument implies that majority of the students did not understand the concept well. The average score was 46% though. In my opinion, outcomes 2.5c needs more instruments for a better evaluation. Later in the recommendations part, I suggest that outcome 2.5 be moved to level 1.

2. The grade distribution was as follows: A: 43 (41.34%), B: 28 (26.92%), C: 14 (13.46%), D: 4 (3.85%), F: 15 (14.42%). The passing rate, including C or better grades, is 85/104 = 81.73%. The passing rate is 89/104=85.58% when A, B, C, and D are considered passing grades. There were a total of 104 students in two sections I taught.
3. Eight of the fifteen students who received F did not attend the final exam.
4. The total initial enrollment was approximately 112 students in two sections. Eight students dropped the class before the end of the semester.

Recommendations:

1. I recommend removal of outcome 2.5c to level 1. The topic is covered in the next course.
2. While the dropping and failing rates are typical, there is a scope of improvement by reducing these rates. A recommendation is to check if there is a correlation between the dropped or failed students with their grades in earlier courses in Computer Science. Possible interventions include mandatory tutoring for students and membership in programming clubs who do not do well at the beginning of the semester. TAs, peer leaders, and the Head TA identified the students at risk in the Fall of 2018 semester but not all the students who ended up with D and F took the advantage of the additional support the department provided. Counselling from the department about challenges of the struggling students is another possible direction.
3. TAs with good programming skills and knowledge in data structures should be recruited to assist this course. The course is heavy in programming skill development. Based on my observation, students do better when there are great TAs who are strong PhD students, diligent in helping students, and have strong communication skills. In the Fall 2018 semester, TAs were not able to grade the lab assignments on time. I think, many of the TAs will benefit from time management training so that they can do well both in teaching and in their own studies.
4. Over the past few years, I modified my in-class teaching methods a bit. My current style involves complex coding using desktop version of Java Virtual Machine and projecting my computer on a large screen. In the classroom, I speak out loud my thoughts during solving a programming problem. Many students personally told me after classes that seeing someone code and hearing out the thought process of a programmer helped them understand critical concepts. One of my recommendations is to keep continuing the practice of solving problems using extensive programming.

Recommendations from previous reports and actions taken:

From previous report: (1) Help students at risk, (2) TAs with good programming skills and knowledge in data structures should be recruited to assist this course.

Actions Taken:

- The TAs and the Head TA offered students at risk additional support.
- The department has started extensive training for TAs.

CS 2401 Elementary Data Structures
CQI Course Review – Fall 2018

Martine Ceberio

1. Course Description

This is the second course for students majoring in Computer Science. Students will learn about fundamental computing algorithms including searching and sorting; recursion; elementary abstract data types including linked lists, stacks, queues and trees; and elementary algorithm analysis.

Knowledge and Abilities Required Before the Students Enter the Course: Students are assumed to be comfortable programming in Java. Students should be able to code basic arithmetic expressions, define simple classes, use strings, code loops and conditional statements, write methods, create objects from classes, invoke methods on an object, perform basic text file input and output, and use arrays.

Prerequisite: CS 1301 and CS 1101 with a grade of C or better in both.

Textbook: Online text: Elementary Data Structures/Algorithms, zybooks, available at zybooks.zyante.com.

2. Learning Outcomes

Level 1: Knowledge and Comprehension: Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to:

1. Explain the concept of polymorphism

Level 2: Application and Analysis: Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:

1. Describe, implement, and use the following concepts:
    a. classes, subclasses, and inheritance
    b. encapsulation and information hiding
2. Describe, implement, and use the following algorithms:
    a. sequential and binary search
    b. quadratic and O(n log n) sorting
    c. string manipulation and parsing
3. Describe and trace computer representation and memory allocation of:
    a. integers, real numbers, arrays and objects
    b. methods, including recursive methods and the use of activation records
4. Use basic notions of algorithm complexity:
    a. use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm
    b. determine the best, average and worst-case behaviors of a simple algorithm
    c. assess basic time and space trade-offs in algorithms
5. Use recursion and iteration as problem solving techniques

Level 3: Synthesis and Evaluation: Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course, students will

be able to:

1. Design and implement solutions to computational problems using the following data structures:
    a. multi-dimensional arrays;
    b. lists implemented as arrays or linked lists;
    c. stacks;
    d. queues;
    e. binary trees and binary search trees.

3. Course Approach

4. Observations and Instruments Mapping to Outcomes

There were 13 students originally registered for the class at the time of the first day of classes: 7 were female students (53.8%).

- No student dropped.
- By the end of the semester:
    - 9 passed the class (69.2%): 2 with an A, 4 with a B, 3 with a C
    - 4 failed the class: 3 with a D, 1 with an F. The student who got an F actually had several family situations (medical, including herself) and is in the process of withdrawing entirely from fall semester.
    - Out of the 7 female students: 4 passed the class and 3 failed (including the student with the medical situation).

It is important to know the following about the students in this section of CS2401. First, this section was exceptionally open to allow an overflow of students to take this course (there was a shortage of space in the other 2 sections). The decision to open this extra section was made the day before the semester started. As a result, most of the students who ended up in my section were students who registered late. In particular, these were mostly probation students. The average GPA of the students in my section was barely 2.0 and 7 of my students had already failed this class at least 1 (some had failed it 3 times, and /or withdrawn multiple times).

Despite this challenge, no student (besides the one with serious medical situations) dropped and the attendance throughout the semester was over 90%. Unfortunately, not all of them were able to make it, but I have high hopes that they will if they keep dedicating themselves as they have throughout the fall semester.

Students were evaluated throughout the semester via quizzes (mostly online, using Socrative.com), bi-weekly homework assignments, three mid-term exams and a comprehensive final exam. Students were also evaluated based on their attendance and participation in class and lab.

CS2401 OUTCOMES MAPPING

	Total	Exams	MT1	MT2	MT3	Final	Quizzes	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Level 1: knowledge and comprehension																		
1. Explain the concept of polymorphism																		
Level 2: Application and Analysis																		
1. Describe, implement, and use the following concepts:	Assessed	X	X			Q4, 5, 6, 7: **66%**												
a. classes, subclasses, and inheritance	Assessed	68%	Q1, 7: **68%**			Q4, 5, 6, 7: **66%**												
b. encapsulation and information hiding	Assessed	68%	Q1, 7: **68%**			Q4, 5, 6, 7: **66%**												
2. Describe, implement, and use the following algorithms:	Assessed	X		X	X													
a. sequential and binary search	Assessed	84.40%			Q7: **80%**	Q10, 11, 12,13,14,15,16: **84.4%**												
b. quadratic and O(n log n) sorting	Assessed	84.40%		Q7: **100%**	Q1: **78.9%**	Q10, 11, 12,13,14,15,16: **84.4%**												
c. string manipulation and parsing																		
3. Describe and trace computer representation and memory allocation of:	Assessed	X	X		X	X												
a. integers, real numbers, arrays and objects	Assessed	73%	Q2, 3: **73%**															
b. methods, including recursive methods and the use of activation records	Assessed	92.30%	Q2, 3: **73%**		Q3: **92.2%**	Q8, 9: **78.9%**												
4. Use basic notions of algorithm complexity:	Assessed	X		X	X	X												
a. use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm	Assessed	84.40%		Q1, 4, 5: **63%**	Q2: **45.5%**	Q10, 11, 12,13,14,15,16: **84.4%**												
b. determine the best, average and worst-case behaviors of a simple algorithm	Assessed	84.40%		Q1, 4, 5: **63%**	Q7: **80%**	Q10, 11, 12,13,14,15,16: **84.4%**												
c. assess basic time and space trade-offs in algorithms	Assessed	63%		Q1, 4, 5: **63%**														
5. Use recursion and iteration as problem solving techniques	Assessed	78.90%		Q2, 3, 6: **87%**		Q8, 9: **78.9%**												
Level 3 Outcomes: Synthesis and Evaluation																		
1. Design and implement solutions to computational problems using the following data structures:	Assessed	X	X	X	X	X												
a. multi-dimensional arrays;	Assessed	82.60%	Q1, 4, 5, 6: **76.9%**			Q1, 2, 3: **82.6%**												
b. lists implemented as arrays or linked lists;	Assessed	66%	Q7: **50%**	Q2: **91.1%**	Q4: **35.6%**	Q4, 5, 6, 7: **66%**												
c. stacks;	Assessed	54%			Q4: **35.6%**	Q17, 18, 19: **54%**												
d. queues;	Assessed	54%			Q5: **62.2%**	Q17, 18, 19: **54%**												
e. binary trees and binary search trees.	Assessed	95.60%			Q6, 7: **86%**	Q20, 21, 22, 23, 24,25,26: **95.6%**												

Following is the mapping of the lab assignments to the outcomes:

	Labs	Arrays	2D Arrays	Array & Objects	Linked Lists	Linked Lists	Recursion	Time Complexity	Stacks	Queues
Level 1: knowledge and comprehension										
1. Explain the concept of polymorphism										
Level 2: Application and Analysis										
1. Describe, implement, and use the following concepts:										
a. classes, subclasses, and inheritance										
b. encapsulation and information hiding										
2. Describe, implement, and use the following algorithms:										
a. sequential and binary search										
b. quadratic and O(n log n) sorting										
c. string manipulation and parsing										
3. Describe and trace computer representation and memory allocation of:										
a. integers, real numbers, arrays and objects										
b. methods, including recursive methods and the use of activation records										
4. Use basic notions of algorithm complexity:										
a. use Big-O notation to express the best-, average- and worst-case behaviors of an algorithm										
b. determine the best, average and worst-case behaviors of a simple algorithm										
c. assess basic time and space trade-offs in algorithms										
5. Use recursion and iteration as problem solving techniques										
Level 3 Outcomes: Synthesis and Evaluation										
1. Design and implement solutions to computational problems using the following data structures:										
a. multi-dimensional arrays;										
b. lists implemented as arrays or linked lists;										
c. stacks;										
d. queues;										
e. binary trees and binary search trees.										
Overal performance		**94.00%**	**81.40%**	**73.40%**	**80.75%**	**80.60%**	**79.70%**	**80.80%**	**80.10%**	**88.85%**

All outcomes were covered in classes and/or in labs.

Regarding the performance of students in the above formal assessments of outcomes, here is a summary:

- **Exams:** Some outcomes were not met in the exams. This is the case of objects and classes (66%), as well as linked lists (66%), stacks and queues (54%). All other outcomes were attained. Given the nature of my group of students (most were on probation), I observed that most were not good exam takers: lack of time management, not understanding the questions and answering off-topic, etc. However, they performed much better on labs (with more time to ask questions and think it through), in which they demonstrated understanding of concepts that they failed at in exams. In classwork, they also were able to demonstrate very good insight in these areas as we practiced a lot with multiple variations of linked lists, stacks, queues.
- **Labs:** Overall all outcomes were attained. Although the performance on the lab on OOP earlier in the semester was low, they made up for it later in the semester when working on labs not focused on OOP but using OOP.

5. Analysis by Instructor:

- **Attendance and dedication:** I was very impressed by the dedication of my students. Attendance was almost perfect throughout the semester and the rate at which students completed their homework was also almost atypical (most students completed 100% of most homework assignment on the online textbook).
- Given the nature of my group of students, I believe that the passing rate is fine. The setting of this class was ideal for these students: smaller group with a lot of hands on activities, a lot of participation, manipulating physical objects to exemplify algorithms and/or to come up with new algorithms. The group was very engaged during the whole semester.
  What I observed is that students lit up when recognized, acknowledged, and encouraged. That's another reason why the smaller group setting worked better I think. One student in particular, whom I had had in a previous course and who had failed the course, was very dedicated, showing a completely different work pattern than previously: she completed all the work that was assigned to her in lecture. Sadly, she did not pass, mostly because she could not attend the labs and almost dropped out of it. This illustrates the difference between a small environment (the lecture) and a large lab.
  Recommendation: We should keep the experiment going on with smaller groups of struggling students and see if we can increase their retention and success this way.
- Overall, outcomes were met (either in exams or in labs). Students developed very good work ethics and habits.
- Online textbook: I used an online textbook, which allowed me to track my students' reading and homework completion. This was very useful in keeping connected to my students and offer tailored help. Recommendation: We should keep using an online textbook with reading and homework tracking.
- Grading: In my grading, I focused on acquisition of skills. As a result, even if a student failed earlier exams, there was always a chance to make up for it in subsequence exams, provided that the failed topics were assessed again. I have been practicing this in CS1 for 4 years and I find it inspiring for our students: it keeps them working. In particular, there is no such thing as "I cannot pass this class even if I get 100 at the final because I failed all previous 3 exams". Similarly, students may fail to submit a lab or 2 because of circumstances outside of their campus commitments: I believe that we should allow students to make up for missed labs with subsequent

ones if they demonstrate mastery of the topics. Professionalism and timeliness matter and they should be taken into account in our overall assessments of students, but they should not prevent us from passing a student if he or she demonstrates mastery of skills.
Recommendation: adopting this way of assessing students since it is more attuned to our students' life circumstances (heavy work load, children, family commitments, etc.).

- Sequence of topics: I would like to revisit the order in which topics are introduced in this course as the sequence did not make a lot of sense and confused students more often than not, forcing me to "open parentheses" and teaching ahead. In particular, as an example, I would recommend covering binary trees right after the review on linked lists, and keep stacks and queues for the end.

Recommendations:

1. We should try to accommodate struggling students in smaller groups and see if we can increase their retention and success this way.
2. We should keep using an online textbook with reading and homework tracking.
3. We should adopt a way of assessing students based on skill acquisition since it is more attuned to our students' life circumstances (heavy work load, children, family commitments, etc.).
4. We should reconsider the sequence in which topics are introduced in this course to make it flow better for students.

Recommendations from previous reports and actions taken:

In the last report, the following was brought up: (1) Help students at risk, (2) TAs with good programming skills and knowledge in data structures should be recruited to assist this course.
Both topics were addressed. TAs and the rest of instructional team members follow a very well structured training program throughout the semester. A head TA is also in charge of visiting labs and administering students' surveys to better understand how well we are performing and adjusting on the fly if needed.

CS 2302 Data Structures
CQI Course Review – Fall 2018 – Diego Aguirre

1. Course Description

Data Structures is the third and final course in the fundamental computer science sequence. Students will learn about fundamental data structures and analysis and design of algorithms.

Knowledge and Abilities Required Before the Students Enter the Course:
Students entering this course are expected to 1) possess fundamental problem-solving skills; 2) implement solutions to computing problems in a high-level programming language; and 3) know about elementary data structures (lists, stacks, and queues). Students must also be familiar with topics from Discrete Math.

Prerequisites: MATH 2300 (Discrete Math) and CS 2401 (Elementary Data Structures and Algorithms)

Textbook:
Online: Data Structures Essentials (Python) - zyBooks

2. Learning Outcomes

Level 3: Synthesis and evaluation:
Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course students will be able to:

1. Given a problem, judge which data structures are required to solve it efficiently and justify the selection.
2. Given a non-recursive algorithm examine its loop structure, assess its asymptotic running time, and express it using big-O notation.
3. Given a recursive algorithm, examine its structure, formulate and solve a recurrence equation defining its running time, and express it using big-O notation.
4. Design and implement solutions to computational problems based on iteration and recursion.
5. Trace the behavior of non-trivial methods and algorithms.

Level 2: Application and analysis:
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details.
Upon successful completion of this course, students will be able to:

1. Describe, implement, and use the following data structures:
    a) Heaps
    b) Hash tables

c) Balanced trees
d) Graphs
e) Disjoint set forests

2. Describe, implement, and apply the following graph algorithms:
   a) Connected components
   b) Breadth-first search
   c) Depth-first search
   d) Topological sorting
   e) Minimum spanning trees (Kruskal's and Primm's)
   f) Single-source shortest paths (Dijkstra's algorithm)

3. Trace the behavior of recursive programs using activation records.
4. Reason about the running times of algorithms in relation to the size of their inputs.

Level 1: Knowledge and comprehension:
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to:

1. Identify and explain the following algorithm design techniques:
   a) Greedy algorithms
   b) Divide and conquer
   c) Dynamic programming
   d) Backtracking

2. Explain the concept of NP completeness.
3. Explain the utility of randomized algorithms.

3. Observations and Instruments
Two sections of this class were offered. Information about both sections is presented below.

Section A: *10:30am – 11:50am*

- Number of students: 60 (This number was calculated after the first month of classes)
- Number of female students: 10 (16.67%)
- From the 60 students, only one dropped; 7 students stopped showing up (did not take the final exam)
- The distribution of grades was the following (including all 59) students:
  - A – 26
  - B – 10
  - C – 12
  - D – 1
  - F – 10

- Two female students did not pass the class (they were part of the group of students that stopped showing up); A – 4; B – 2, C – 2; F – 2

Section B: *1:30pm – 2:50pm*

- Number of students: 34 (This number was calculated after the first month of classes)
- Number of female students: 3 (8.8%)
- 4 students stopped showing up (did not take the final exam)
- The distribution of grades was the following (including all 34) students:
  - A – 15
  - B – 9
  - C – 3
  - D – 3
  - F – 4
- All females passed the class; A -2; B-1

Students were evaluated throughout the semester via zyBooks quizzes, homework assignments, three partial exams and a comprehensive final examination. In addition to this, students were assigned 7 labs where they had to apply the concepts learned in class.

Changes made to the course

Multiple changes were made to the course. The following list presents all of these changes along with justifications for such modifications.

- High-level programming language changed from Java to Python.

  Python is quickly becoming an important programming language. Modern libraries used in fields like data science and cyber-security are being written in Python. Students will benefit from learning about multiple programming languages as they advance in the program. The concepts taught in this course require basic coding knowledge. It is relatively straightforward for students to transition to Python from Java.

- Each lab had different versions; students were allowed to select the version they wanted to work on.

  To motivate students and provide them with a sense of ownership, multiple versions of each lab were written. All versions targeted the same learning outcomes, but the problem to be solved changed. Students were allowed and encouraged to select the version of the lab that appeal to them the most. Additionally, each lab came with an extra-credit section where students were given the opportunity to solve more challenging problem within the same context.

- Every lab required students to code review the projects (labs) submitted by others

  Code reviews are an essential practice in modern software development. This practice not only allows students to learn about other people's code, but helps them develop good coding practices, find bugs, improve coding skills, and develop stronger problem-solving skills.

- Exams were graded automatically using unit tests.

  All exams required students to use their laptops. Coding problems were given to students at test time. Students were tasked with completing partial implementations of code provided by the instructor. Students were also given a set of unit tests to assess their code as they were writing it. That is, students were encouraged to code, run the provided unit tests, and fix their code if necessary. The code was not trivial, so students really needed to understand the material to get a good grade. Exams also had multiple-choice and fill in the blank questions.

- All reading material and quizzes came from zyBooks

  The zyBook used in class covered the same concepts that have been traditionally covered in previous iterations of the class. This made the transition for the instructor easy. Personally, I think zyBooks has done a great job creating material that is easy to digest and learn from. All quizzes were assigned through zyBooks and automatically graded by the system.

- Students were allowed to re-take exams 1 and 2

  Students were given the opportunity to make-up for the first and second partial exams. These two make-up exams had the same number of questions as the original exams, and each question targeted the same learning outcome(s). The idea was to motivate students to continue studying even if they did poorly in one of the exams.

The following table shows how each learning outcome maps to an exam's question or lab assignment.

Learning Outcome	**Exam 1 / Make-Up Exam 1**	**Exam 2 / Make-Up Exam 2**	**Exam 3**	**Final Exam**	**Labs**
L1.1.a			Q13	Q42	
L1.1.b			Q10		
L1.1.c			Q5, Q12	Q38	Lab 7
L1.1.d				Q42	
L1.2			Q14	Q41	
L1.3			Q11	Q42	

L2.1.a		Q5, Q15, Q16, Q17, Q18		Q23, Q25, Q31, Q32	Lab 5
L2.1.b		Q4, Q11, Q12, Q13, Q14		Q23, Q24, Q29, Q30	Lab 4
L2.1.c		Q1, Q2, Q3, Q7, Q8, Q9, Q10		Q21, Q22, Q23, Q27, Q28	Lab 3
L2.1.d			Q6, Q7, Q15, Q16, Q17, Q18, Q19, Q20, Q21, Q22	Q39, Q43, Q44, Q45, Q46	Lab 6
L2.1.e		Q6, Q19, Q20, Q21, Q22		Q23, Q26, Q33, Q34	
L2.2.a			Q6, Q7	Q39	Lab 6
L2.2.b			Q2, Q8	Q36	
L2.2.c			Q3, Q9	Q40	
L2.2.d			Q1	Q35	Lab 6
L2.2.e			Q6, Q7	Q39	
L2.2.f			Q4	Q37	
L2.3	Q32, Q33, Q34			Q15, Q16, Q17	
L2.4	Q24, Q25, Q26, Q27, Q28, Q29			Q11, Q12, Q13	Lab 1 – Lab 7
L3.1			Q8, Q9	Q40	Lab 3’
L3.2	Q4, Q5,Q6, Q7, Q8, Q9, Q10, Q11			Q1, Q2, Q3, Q4	
L3.3	Q12, Q13, Q14, Q15, Q16, Q17, Q18, Q19, Q20, Q21, Q22, Q23, Q30, Q31			Q5, Q6, Q7, Q8, Q9, Q10, Q14	
L3.4					Lab1 – Lab7
L3.5	Q35, Q36, Q37, Q38,	Q1, Q2, Q3, Q4, Q5, Q6	Q1, Q2, Q3, Q4, Q5	Q18, Q19, Q20, Q24,	Lab 1

	Q39, Q40, Q41, Q42			Q25, Q26, Q35, Q36, Q38	

Exam 1 (left) and its make-up (right) consisted of 39 questions. The following tables show the average score per question (0 to 1)

Question Number	Section A - 10:30am	Section B - 1:30pm	Two Sections
1	0.8666666667	0.8571428571	0.8631205674
2	0.3666666667	0.2	0.3046099291
3	0.8833333333	0.7714285714	0.8416666667
4	0.8333333333	0.8571428571	0.8421985816
5	0.5833333333	0.7714285714	0.6533687943
6	0.8333333333	0.8571428571	0.8421985816
7	0.9166666667	0.8	0.8732269504
8	0.6	0.7142857143	0.6425531915
9	0.85	0.9428571429	0.8845744681
10	0.85	0.8	0.8313829787
11	0.6166666667	0.5714285714	0.599822695
12	0.65	0.6571428571	0.6526595745
13	0.7	0.7142857143	0.7053191489
14	0.6333333333	0.6285714286	0.6315602837
15	0.4833333333	0.7142857143	0.5693262411
16	0.5833333333	0.6	0.5895390071
17	0.8666666667	0.8285714286	0.8524822695
18	0.8166666667	0.8857142857	0.8423758865
19	0.8	0.9142857143	0.8425531915
20	0.7333333333	0.7142857143	0.7262411348
21	0.6833333333	0.9142857143	0.7693262411
22	0.6	0.8	0.6744680851
23	0.6666666667	0.7142857143	0.6843971631
24	0.2166666667	0.5142857143	0.3274822695
25	0.5333333333	0.7714285714	0.6219858156
26	0.5833333333	0.7142857143	0.6320921986
27	0.6333333333	0.6285714286	0.6315602837
28	0.4166666667	0.3428571429	0.3891843972
29	0.4333333333	0.5142857143	0.4634751773
30	0.55	0.7714285714	0.6324468085
31	0.5166666667	0.7142857143	0.590248227
32	0.3833333333	0.4571428571	0.4108156028
33	0.6	0.6285714286	0.6106382979
34	0.5	0.6	0.5372340426
35	0.5833333333	0.7142857143	0.6320921986
36	0.5666666667	0.6857142857	0.6109929078
37	0.4166666667	0.4857142857	0.4423758865
38	0.4166666667	0.5142857143	0.4530141844
39	0.4333333333	0.3428571429	0.3996453901

Question Number	Two Sections
1	0.5714285714
2	0.5714285714
3	0.9428571429
4	0.5142857143
5	0.7714285714
6	0.7142857143
7	0.7428571429
8	0.6285714286
9	0.8571428571
10	0.8571428571
11	0.8571428571
12	0.9142857143
13	0.9714285714
14	0.9714285714
15	0.7428571429
16	0.7142857143
17	0.8285714286
18	0.8857142857
19	0.7714285714
20	0.4571428571
21	0.8285714286
22	0.6571428571
23	0.8571428571
24	0.3714285714
25	0.4857142857
26	0.4571428571
27	0.8857142857
28	0.8
29	0.6
30	0.6571428571
31	0.6285714286
32	0.4571428571
33	0.2857142857
34	0.4
35	0.8
36	0.6857142857
37	0
38	0.5428571429
39	0.4571428571

Exam 2 (left) and its make-up (right) consisted of 22 questions. The following tables show the average score per question (0 to 1)

Question Number	Section A - 10:30am	Section B - 1:30pm	Two Sections
1	0.7111111111	0.6206896552	0.6756756757
2	0.5333333333	0.275862069	0.4324324324
3	0.7555555556	0.6896551724	0.7297297297
4	0.4444444444	0.724137931	0.5540540541
5	0.7777777778	0.7931034483	0.7837837838
6	0.4222222222	0.4482758621	0.4324324324
7	0.2444444444	0.4482758621	0.3243243243
8	0.4666666667	0.4137931034	0.4459459459
9	0.3333333333	0.4827586207	0.3918918919
10	0.3333333333	0.4137931034	0.3648648649
11	0.3333333333	0.3448275862	0.3378378378
12	0.6	0.6206896552	0.6081081081
13	0.7555555556	0.724137931	0.7432432432
14	0.6	0.6551724138	0.6216216216
15	0.2	0.2068965517	0.2027027027
16	0.3777777778	0.3448275862	0.3648648649
17	0.3555555556	0.3103448276	0.3378378378
18	0.2222222222	0.2068965517	0.2162162162
19	0.5111111111	0.6551724138	0.5675675676
20	0.4222222222	0.4827586207	0.4459459459
21	0.1777777778	0.2068965517	0.1891891892
22	0.2444444444	0.2413793103	0.2432432432

Question Number	Section A - 10:30am	Section B - 1:30pm	Two Sections
1	0.7692307692	0.6206896552	0.7172413793
2	0.7435897436	0.275862069	0.5798850575
3	0.8974358974	0.6896551724	0.8247126437
4	0.6666666667	0.724137931	0.6867816092
5	0.7948717949	0.7931034483	0.7942528736
6	0.1794871795	0.4482758621	0.2735632184
7	0.641025641	0.4482758621	0.5735632184
8	0.7435897436	0.4137931034	0.6281609195
9	0.5641025641	0.4827586207	0.5356321839
10	0.8205128205	0.4137931034	0.6781609195
11	0.9230769231	0.3448275862	0.7206896552
12	0.8974358974	0.6206896552	0.8005747126
13	0.6923076923	0.724137931	0.7034482759
14	0.6666666667	0.6551724138	0.6626436782
15	0.358974359	0.2068965517	0.3057471264
16	0.8717948718	0.3448275862	0.6873563218
17	0.641025641	0.3103448276	0.5252873563
18	0.8205128205	0.2068965517	0.6057471264
19	0.7179487179	0.6551724138	0.6959770115
20	0.641025641	0.4827586207	0.5856321839
21	0.7435897436	0.2068965517	0.5557471264
22	0.3333333333	0.2413793103	0.3011494253

Exam 3 consisted of 22 questions. The following table shows the average score per question (0 to 1)

Question Number	Section A - 10:30am	Section B - 1:30pm	Two Sections
1	0.8272727273	0.8888888889	0.8503787879
2	0.696969697	0.7525252525	0.7178030303
3	0.3545454545	0.3888888889	0.3674242424
4	0.5090909091	0.6333333333	0.5556818182
5	0.8671717172	0.7575757576	0.8260732323
6	0.9272727273	0.9090909091	0.9204545455
7	0.8727272727	0.9090909091	0.8863636364
8	0.7636363636	0.7878787879	0.7727272727
9	0.7454545455	0.8181818182	0.7727272727
10	0.8	0.7575757576	0.7840909091
11	0.8727272727	0.8787878788	0.875
12	0.4909090909	0.7272727273	0.5795454545
13	0.6727272727	0.7575757576	0.7045454545
14	0.6363636364	0.6363636364	0.6363636364
15	0.5818181818	0.7575757576	0.6477272727
16	0.5090909091	0.6666666667	0.5681818182
17	0.5272727273	0.4848484848	0.5113636364
18	0.2181818182	0.3333333333	0.2613636364
19	0.5272727273	0.6363636364	0.5681818182
20	0.4363636364	0.6666666667	0.5227272727
21	0.5272727273	0.6666666667	0.5795454545
22	0.2181818182	0.3939393939	0.2840909091

The final exam consisted of 46 questions. The following table shows the average score per question (0 to 1)

Question Number	Section A - 10:30am	Section B - 1:30pm	Two Sections
1	0.9807692308	0.9677419355	0.9759036145
2	0.9615384615	0.9677419355	0.9638554217
3	0.5576923077	0.4838709677	0.5301204819
4	0.8076923077	0.7419354839	0.7831325301
5	0.8653846154	0.8709677419	0.8674698795
6	0.7115384615	0.7096774194	0.7108433735
7	0.8076923077	0.8387096774	0.8192771084
8	0.9230769231	0.935483871	0.9277108434
9	0.7307692308	0.7741935484	0.7469879518
10	0.9038461538	0.8387096774	0.8795180723
11	0.7884615385	0.8709677419	0.8192771084
12	0.4230769231	0.4516129032	0.4337349398
13	0.9423076923	0.8709677419	0.9156626506
14	0.9615384615	0.7741935484	0.8915662651
15	0.9230769231	0.8064516129	0.8795180723
16	0.7115384615	0.7096774194	0.7108433735
17	0.8461538462	0.7741935484	0.8192771084
18	0.7692307692	0.8387096774	0.7951807229
19	0.75	0.7741935484	0.7590361446
20	0.6153846154	0.7419354839	0.6626506024
21	0.9423076923	0.8709677419	0.9156626506
22	0.8653846154	0.7419354839	0.8192771084
23	0.8461538462	0.8709677419	0.8554216867
24	0.8269230769	0.8064516129	0.8192771084
25	0.9423076923	0.8387096774	0.9036144578
26	0.6538461538	0.3870967742	0.5542168675
27	0.7307692308	0.935483871	0.8072289157
28	0.8653846154	0.9032258065	0.8795180723
29	0.9807692308	0.9677419355	0.9759036145
30	0.8076923077	0.8387096774	0.8192771084
31	0.5769230769	0.6774193548	0.6144578313
32	0.6153846154	0.6774193548	0.6385542169
33	0.9038461538	0.9677419355	0.9277108434
34	0.6346153846	0.7741935484	0.686746988
35	0.8429487179	0.876344086	0.8554216867
36	0.8397435897	0.8817204301	0.8554216867
37	0.8245192308	0.689516129	0.7740963855
38	0.921875	0.8366935484	0.890060241
39	0.8076923077	0.7741935484	0.7951807229
40	0.8076923077	0.8709677419	0.8313253012
41	0.4615384615	0.1612903226	0.3493975904
42	0.75	0.6774193548	0.7228915663
43	0.8076923077	0.8709677419	0.8313253012
44	0.7692307692	0.7096774194	0.7469879518
45	0.9230769231	0.935483871	0.9277108434
46	0.7692307692	0.7741935484	0.7710843373

The following table shows the averages (0 to 1) of all lab assignments (counting only those students that submitted the labs and not including late submission penalties):

	Section A – 10:30am	**Section B – 1:30pm**
Lab 1	0.83	0.85
Lab 2	0.79	0.72
Lab 3	0.92	0.92
Lab 4	0.89	0.91
Lab 5	0.92	0.90
Lab 6	0.93	0.84
Lab 7	0.93	0.97

The following table shows the averages (0 to 1) of all exams (counting only those students that submitted the labs and not including late submission penalties):

	Section A – 10:30am	**Section B – 1:30pm**
Max (Exam 1, Exam 1 – Make Up)	0.83	0.86
Max (Exam 2, Exam 2 – Make Up)	0.77	0.77
Exam 3	0.63	0.74
Final Exam	0.79	0.81

4. Analysis by the instructor

Some observations of the information presented above:

- The exam averages were better than in previous iterations of the course, except for exam 3. More coding questions were added to that exam, making it more challenging than in previous semesters.
- Students liked the idea of having multiple versions of the same lab. Students stated that they liked to see different contexts in which the same data structure can be used to solve a problem. The lab grades were better than expected.
- Code reviews were mandatory, but they were just a 'binary' part of the lab grade. As long as they provided feedback to a teammate, they would get the part of the lab done. Because of this, the quality of some reviews was poor.
- Even though the final exam was comprehensive and long, students did great.
- The areas where the students struggled the most are: running time analysis (big-O for both recursive and non-recursive algorithms), hash tables, and disjoint set forests.
- Based on the final exam results, and the average of lab grades, most outcomes were attained. The outcomes mentioned in the previous bullet point were the weakest

- Overall, the skills that labs aimed to tackle were mostly attained.
- The number of students that did not pass the class (combining the two sections) was 18, which represents 19% of the students that took the class.
- Many students stopped showing up without formally dropping the class.
- The use of zyBooks really helped. Students read about the topics covered in class more often, and the quizzes were easy to distribute and grade.This freed our TAs and IAs, which allowed them to help students more.
- The IA did an excellent job, his help was very valuable.
- The change from Java to Python did not seem to have had very negative consequences. Students liked adding a second language to their toolbox. Some struggled, but it was easy to help them since the computational constructs used in class were very basic. ZyBooks also helped with this, as they added chapters from an introductory Python course.
- Students like the fact that they can take make-up exams. Many of the greatly improved their grades because of this.
- Grading coding exams at test time with the use of unit tests was challenging, but overall I think it worked great. Some students submitted code that did not work, which required manually fixing it to correctly assess their work. Students liked that they could bring their computers and take the exam there. They specially liked that unit tests would help them understand how fix their code while allowing them to know their grades at test time.

Recommendations

- The program that supports instructional assistant should continue to be funded. Students benefited from the IA greatly.
- We can do a much better job with code reviews. Given them an official rubric and assessing them more thoroughly is necessary for the idea to work.
- Students actually read the zyBook, we should continue using it.
- Even though not a lot of students dropped the course, many of them stopped showing up. Something must be done.
- Having multiple versions of a lab is great for students, but it's time-consuming for the instructor. We should think about how to proceed.