Department of Computer Science  
2020 Assessment Report

All areas shaded in gray are to be completed by the department/program.  
This document will be posted online and must be accessible electronically (including appendices).

MISSION STATEMENT
The Computer Science Department at University of Montana-Missoula dedicates itself and its resources to the growing utility of computers in research and education, as well as the increased impact of computers on our modern society. Our primary mission is to offer degrees (major, minor, graduate) for students who want to pursue a career in this discipline. Primary objectives of our curriculum include: 1) develop professionally competent and broadly educated computer scientists who wish to pursue professional careers or graduate studies, 2) mentor students and provide them with opportunities to engage in CS research, and 3) teach students how to think computationally and engage in problem solving and critical analysis.

DEPARTMENT ALIGNMENT WITH PRIORITIES FOR ACTION
After listing each departmental objective, indicate which of the five Priorities for Action the objective supports. In this section, you may also briefly describe any innovative or noteworthy programs/initiatives that support the Priorities for Action.

The Computer Science Department successfully proposed a new curriculum with three concentrations of study during the 2019-2020 school year. The curriculum was updated and changed for the betterment of students, the institution and the Montana workforce. The changes were guided by meetings with our advisory board of computing professionals, an exit survey given to all graduating seniors, in-class assessments at all levels, the University wide “Priorities for Action”, and our reading of the changes taking place in the computing sector of the economy.

The new curriculum articulates a Computer Science Core of 29 credits, and provides students with three distinct concentrations of study in sought-after areas of Computer Science: Data Science, Software Engineering, and Algorithm Design.

The objectives of the curriculum changes are as follows:
1. increase the quality of the curricula as they are directly responsive to current workforce demands;
2. boost student performance and interest by providing more routes to completion via three distinct concentrations;
3. prepare every graduate to excel in the areas of problem-solving, adaptability, communication, critical thinking, collaboration, creativity, and ethical reasoning; and
4. engage students in interdisciplinary education.

Our curriculum proposal described how these objectives support UM’s Mission and Strategic Opportunities. They also support the Priorities for Action:

1. Place student success at the center of all we do [1-4]
2. Drive excellence and innovation in teaching, learning and research [1-4]
4. Partner with Place [1,4]
5. Proudly tell the UM story [via capstone projects and work in the community]
Some noteworthy curricular features are as follows:

1. Three distinct concentrations of study on top of a common core were designed to improve student interest and retention, and to meet evolving industry needs.

2. The introductory sequence, CSCI 135/136 Fundamentals of Computer Science I/II, was replaced with a three-course sequence CSCI 150, 151, 152 Introduction to Programming (150), and Interdisciplinary Computer Science I, II (151,152). The goal of this change is to increase retention and to better prepare students for upper-division coursework by devoting increased time (3 vs. two semesters of introductory courses) to computing fundamentals.

3. A flexible, modular design providing UM students opportunities to learn computational skills but not necessarily a computer science degree through micro-credentials including a programming certificate and a well-articulated minor in CS.

4. A newly approved Ph.D program, which will generate better faculty recruitment and retention, provide more assistantships, create undergraduate research opportunities and contribute to a more robust scholarly community.

5. A renewed commitment to interdisciplinary research and education; our department prides itself on the research collaborations its faculty have across the campus and the world, and how those collaborations inform our teaching and guide the student experience at all levels.

6. A flexible, modular design that provides students the ability to build upon foundational knowledge in computational thinking and programming and expand further into areas of particular interest to themselves and their future goals.

STUDENT LEARNING GOALS and MEASUREMENT TOOLS

The accreditation body for computer science and engineering programs is the Accreditation Board for Engineering and Technology (ABET.) Although our programs discontinued accreditation through ABET in 2018, the Computer Science Department continues to use the well regarded ABET outcomes for assessment purposes. The outcomes are as follows:

1. Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.

2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline.

3. Communicate effectively in a variety of professional contexts.

4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

5. Function effectively as a member or leader of a team engaged in activities appropriate to the program’s discipline.

6. Apply computer science theory and software development fundamentals to produce computing-based solutions.
<table>
<thead>
<tr>
<th>Student Learning Goals</th>
<th>Classroom Assessments and FCARs</th>
<th>Capstone</th>
<th>Exit Survey</th>
<th>Advisory Board</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>3. Communicate effectively in a variety of professional contexts.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>5. Function effectively as a member or leader of a team engaged in activities appropriate to the program’s discipline.</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>6. Apply computer science theory and software development fundamentals to produce computing-based solutions.</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### RESULTS and MODIFICATIONS – Overall Goals Based on New Curriculum

<table>
<thead>
<tr>
<th>Curricular Change and Assessment Goals</th>
<th>Modifications made to enhance learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>The introductory fundamentals sequence, CSCI 135-136 was expanded to three courses: 150-151-152. By spreading the material into 3 courses, we’re able to accommodate students with no programming background, yet still provide a pace that allows students to successfully build a foundation. We also moved some material that was previously in 232 down into 152, which allows 232 to also slow down and establish a stronger base. Overall, we spread 3 tightly-packed course’s worth of material across 4 courses, with the aim of improving mastery and retention. As many in the sciences, mathematics, and business are interested in developing programming skills, there is now broad demand for CSCI 150. Having such a broad and diverse cohort in CSCI 150 is helping to create a more inclusive culture in computer science.</td>
<td>Faculty will define ways to determine whether students are better prepared for CSCI 232 than they were in past years. We will focus on the relative retention rates between each of the fundamentals courses and onwards into our sophomore sequence; special consideration will be given to retention of women and other groups under-represented in computer science.</td>
</tr>
<tr>
<td>Increase gender diversity in the program.</td>
<td>Measure retention of female students through the fundamentals courses: 150, 151, 152 and 232. Interview female students who are retained in the program to determine what is working for them.</td>
</tr>
<tr>
<td>Increase the number and quality of internships reported by graduating seniors.</td>
<td>The Exit Survey is used to gather internship data from students. Consider means of gathering data from intern employers. Provide more incentive for students to participate in internships while in school. One such incentive is to change the program to require internship OR research OR capstone. Begin to promote internships early and often, i.e. in the Careers class.</td>
</tr>
</tbody>
</table>
FUTURE PLANS FOR CONTINUED ASSESSMENT

Since this is the start of a new curriculum cycle, the department is taking a holistic look at the way we conduct assessment. This Fall, several faculty meetings were spent revising the mission, reviewing student learning outcomes, discussing how we measure outcomes, and potential changes to our assessment activities. A curriculum map was developed for the CS core and each of the concentrations.

The department’s assessment goals for the future are to:

- Review the ABET learning outcomes and determine whether they should be modified for our major and concentrations of study.
- Develop additional benchmarks and quantitative metrics for measuring learning outcomes; we are particularly interested in skills based assessments stemming from professional examinations that exist in the computer science field.
- Acquire/gather retention data and other means to measure success in terms of recruitment and retention.
- Determine how to best use assessment data to improve the program while students are still in school.
- Continue to use the FCARs to assess lower-level class’ mapping to our learning outcomes.
- Continue to use the Exit Survey and the Advisory Board to inform high level curricular changes and department goals.
- Revise the mission statement.
- Continue assessing learning at the course level, but also consider how to perform pre and post data gathering techniques for a group of courses (e.g., CSCI 150-151-152) as well as at the beginning and end of the degree program.

Continued adjustments are expected as the new classes and concentrations are implemented and assessed, and as more data become available.

APPENDICES

1. Curriculum Map (Excel)
2. Exit Survey Results (pdf)
3. FCARs for CSCI135 and CSCI 361
Curriculum Mapping helps us analyze the path students take through the curriculum, and where/how they master outcomes.

Please consider the classes you teach. For each of the major outcomes, determine the level at which each student should meet each outcome in your classes. Remember, not all classes will meet all outcomes. Identify assignments and activities that allow students to demonstrate their proficiency of the outcome.

Levels (suggested by the Provost's office):  
"I"=Introduced;  
"D"=developed/reinforced, with opportunities to practice;  
"M"=mastery that is demonstrated (often at the senior or exit level);  
"A"=assessment evidence collected

ABET Computer Science Outcomes
1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.  
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.  
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]  
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.  
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
Curriculum Mapping helps us analyze the path students take through the curriculum, and where/how they master outcomes.

Please consider the classes you teach. For each of the major outcomes, determine the level at which each student should meet each outcome in your classes. Remember, not all classes will meet all outcomes. Identify assignments and activities in the class that allow students to demonstrate:

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
ABET Computer Science Outcomes:
1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline.
3. Communicate effectively in a variety of professional contexts (presentation, group discussion, interpersonal, high-quality report, email).
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to
<table>
<thead>
<tr>
<th>3, Comm</th>
<th>4, Ethics</th>
<th>5, Teamwork</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M, A</td>
<td>M, A</td>
<td></td>
</tr>
</tbody>
</table>

Key: “I”=Introduced; “D”=developed/reinforced, with opportunities to practice; “M”=mastery that is demonstrated (often at the senior or exit level); “A”=assessment evidence collected.

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program’s discipline.

3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]

4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

5. Function effectively as a member or leader of a team engaged in activities appropriate to the program’s discipline.
I put "D" for Comm due to emphasis on written answers, but you may disagree?

d
<table>
<thead>
<tr>
<th>Course</th>
<th>1. Analyze</th>
<th>2. Design</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI 181 Web Design &amp; Programming</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>CSCI 322 Advanced Web Application Development</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>CSCI 340 Database Development</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>CSCI 426 Capstone</td>
<td>M, A</td>
<td>M, A</td>
</tr>
<tr>
<td>CSCI 427 Capstone</td>
<td>M, A</td>
<td>M, A</td>
</tr>
<tr>
<td>CSCI 443 User Interface Design or CSCI 400 Digital</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Entrepreneurship **</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

15 upper-division electives

Standardize outcomes?

** These are two very different classes. Should we only use the outcome(s) that they have in common?

** ABET Computer Science Outcomes:  
1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.  
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.  
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.  
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
Questions

3, Comm  4, Ethics  5, Teamwork

M, A  M, A
M, A  M, A

These are two very different classes. Should we only use the outcome(s) that they have in common?

Key: "I"=Introduced; "D"=developed/reinforced, with opportunities to practice; "M"=mastery that is demonstrated (often at the senior or exit level); "A"=assessment evidence collected

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
<table>
<thead>
<tr>
<th>Course</th>
<th>1. Analyze</th>
<th>2. Design</th>
<th>3. Comm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI 432 Advanced Algorithm Development</td>
<td>M</td>
<td>M</td>
<td>D</td>
</tr>
<tr>
<td>CSCI 361 Computer Architecture</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
<tr>
<td>12 cr. upper-division electives</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The remaining courses are Math courses, which we don’t assess.

Key: "I"=Introduced; "D"= developed/reinforced, with opportunities to practice; "M"=mastery that is demonstrated (often at the senior or exit level); "A"=assessment evidence collected

ABET Computer Science Outcomes:
1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
<table>
<thead>
<tr>
<th>4, Ethics</th>
<th>5, Teamwork</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
</tbody>
</table>

(filled by TW; Comm could maybe be "M" - they have significant presentation requirements)

Key: "I"=Introduced; "D"=developed/reinforced, with opportunities to practice; "M"=mastery that is demonstrated (often at the senior or exit level); "A"=assessment evidence collected

1. Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts. [presentation, group discussion, interpersonal, high-quality report, email]
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.
nificant presentation requirements)
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI 477 Simulation</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>CSCI 447 Machine Learning</td>
<td>M</td>
<td>M</td>
<td>D</td>
</tr>
<tr>
<td>CSCI 444 Data Visualization</td>
<td></td>
<td></td>
<td>M</td>
</tr>
<tr>
<td>CSCI 340 Database Development</td>
<td>D</td>
<td>D</td>
<td></td>
</tr>
</tbody>
</table>

6-9 cr. of CS electives

Standardize outcomes?

5-6 Data Science elective choices / only 2-3 are CS

Key: "I"=Introduced; "D"= developed/reinforced, with opportunities to practice; "M"=mastery

**ABET Computer Science Outcomes:**

1. Analyze a complex computing problem and to apply principles of computing and other related disciplines to identify solutions.
2. Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3. Communicate effectively in a variety of professional contexts, such as presentations, group discussions, interpersonal interactions, high-quality reports, and emails.
4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.
5. Function effectively as a member or leader of a team engaged in activities appropriate to the discipline.
4, Ethics  

<table>
<thead>
<tr>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

5, Teamwork

| D |

that is demonstrated (often at the senior or exit level); "A"=assessment evidence collected
What is your immediate career plan after graduation?

12 responses

- 50% Work
- 25% Graduate school
- 16.7% Not sure
- 8.3% Finish my minors in media arts and math.

Work section
What is the name of the company you are starting your job with?
4 responses

Montana Department of Justice
Bain and Company
FAST Enterprises LLC
NULL

Where is this company located (City, State)?
4 responses

Helena, MT
Boston, MA
Centennial, CO
NULL
What will your position be (e.g., programmer, software engineer, etc.)?
4 responses

- Computer Programmer
- Software Engineer
- Implementation Consultant
- Programmer, Front-end, Back-end

What is your starting salary range?
4 responses

- 100K+
- 80-100K
- 60-80K
- 40-60K
- <40K
- Prefer not to answer

75%
25%
How well do you think our program prepared you for your job search?

6 responses

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>50%</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>33.3%</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>16.7%</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

Graduate school section

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many in-State grad schools did you apply for?</td>
<td>6</td>
</tr>
<tr>
<td>How many in-State grad schools were you accepted by?</td>
<td>4</td>
</tr>
<tr>
<td>How many out-of-State grad schools did you apply for?</td>
<td>8</td>
</tr>
<tr>
<td>How many out-of-State grad schools were you accepted by?</td>
<td>8</td>
</tr>
</tbody>
</table>

https://docs.google.com/forms/d/12trxeXWUaBgi1rVdhXG-zXIfE3aBkd5fcDQAqL0ReA/edit?ts=60159657&gxids=7628#responses
What graduate school will you be attending (Name) and where is it located?

4 responses

University of Montana, Missoula

N/A

Hopefully the University of Montana

University of Montana (Maybe)

Will you be a Masters candidate or a PhD candidate?

5 responses

- Masters: 100%
- PhD: 0%
What will you be studying in graduate school?

4 responses

- **Computer Science**: 2 (50%)
- **Mathematics**: 1 (25%)
- **N/A**: 1 (25%)

How well do you think our program prepared you for graduate school?

7 responses

- **1**: 0 (0%)
- **2**: 1 (14.3%)
- **3**: 1 (14.3%)
- **4**: 2 (28.6%)
- **5**: 3 (42.9%)
What is your GPA?
12 responses

3.71
3.5
3.7
3.33
3.87
3.36
3.8
3.32
2.89

Did you receive a Pell grant while at UM?
12 responses

50% Yes
50% No
How many internships did you participate in during your time in our program?

12 responses

- 0: 75%
- 1: 8.3%
- 2: 16.7%

How valuable was your internship experience?

3 responses

- 1: 33.3%
- 2: 66.7%
Did you participate in any undergraduate research while in our program?

12 responses

- Yes: 75%
- No: 25%

Research follow-up

How valuable was your undergraduate research experience?

3 responses

- 1 (33.3%)
- 2 (66.7%)

Satisfaction
Please rate the following aspects of the CS program?

![Bar chart showing ratings for Value of course content, Facilities and Resources, and Faculty / Instructors.]

What were the biggest obstacles to completing your degree (check all that apply)?

12 responses

- Challenging classes: 6 (50%)
- Reconciling class schedules: 5 (41.7%)
- Financial stress: 2 (16.7%)
- Personal issues: 6 (50%)
- There weren't any, it was too easy: 1 (8.3%)
- Relationships with some people induced …: 1 (8.3%)
How likely are you to recommend the CS program at UM to a friend or colleague?

12 responses

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8.3%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>25%</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>50%</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>8.3%</td>
</tr>
</tbody>
</table>

Please provide additional comments on areas of our program that you think are particularly good or that can use improvement.

11 responses

Needs more rigour

I feel that the CS program severely lacks preparation for web development and database work. I was introduced to these concepts during my capstone class, but I needed them much earlier than that.

I’m glad the curriculum is changing; the current one isn’t ideal. I’ve also noted the classes that best prepared me for the outside were classes like Capstone, when I had a client and had to do the things that software engineers would have to do. The coding classes were ok to learn the standards of syntax, but learning more in depth about object orientation would have been a lot more help. Also, we have no classes that actually teach us about available frameworks in comparison to starting a site with no framework and doing it from scratch. That said, the Database Design class really taught me the basics of how to even start a website. The Software Science class I took, we used Ruby and Ruby on Rails, and we were essentially told to “figure it out” with no help. We had to rely on the internet to even guide us. It ended up terrible and ultimately, we still had no idea what we were doing. Yolanda’s Capstone and Database Design classes are the 2 classes that really helped me the most. Travis’s Data structures class really helped lay a foundation of the available ways to store data and their efficiency. Oliver’s Algorithms class left me wondering why that mattered. The class was a great way to teach students why a certain algorithm performed the way it did and prove it, but in the field that I’m headed, do I care? Our main goal is to be able to use the algorithm, not prove it. It’s already been proven, teach
This class is designed to give students a good general understanding of software development and logical reasoning. This course focuses on introducing general programming constructs using the Python programming language. This course introduces all of the following concepts as well as provides a number of hands on opportunities to become proficient in using these tools.

- General Computing Concepts
- Logical Reasoning and Computational Thinking
- Programming Constructs
- Object Oriented Concepts

**Prerequisites:** none

**Income Assessment:**

The students taking this course are typically first or second semester freshman. They may have been exposed to some programming constructs either through CSCI 100, high school programming courses, or self-study, but they are not expected to have much programming experience.

**Modifications made versus previous offering:**

- The class was modified after spring break to be completely online. Videos of lectures were created and posted on Moodle. The class has been tweaked each semester by incorporating more student activity in the classroom. Students are given short programming problems to complete during lectures.

**Course Outcomes Assessment:**

This course consisted of:

- 13 activities with zyBooks. zyBooks is a web platform for learning STEM material. zyBooks has minimal text. It consists of question sets, animations, interactive tools, and embedded homework, so students can learn by doing.
- 11 labs (Instructor and teaching assistant provided as much help as necessary for students to work through the labs).
- 13 homework assignments (Minimal assistance by instructor and teaching assistant so students could develop his or her problem solving skills.)
- 3 exams
Activities, labs, homework and exams provided what was necessary for students to learn the following outcomes:

1. Understand the basic components of a computer and how it works
2. Understand data types
3. Create graphical programs
4. Implement appropriate looping and control structures to solve problems
5. Create test cases for programs written
6. Read from files, iterate through the file and manipulate the data within the file
7. Analyze a problem, and identify and define the computing requirements appropriate to its solution
8. Use current techniques, skills, and tools necessary for computing practice

The following is an assessment of an assignment given during the last third of the semester in Spring 2020, Fall 2019, Fall 2018. The assignment tests outcomes (2,4,6,7,8) above. Students have approximately 1 week to complete the assignment and can submit their program unlimited times until the due date.

**Spring 2020**
Reflection:

The most encouraging statistic from the above data is the number of students submitting work. Student tend to give up as the material gets more difficult. The additional in class activities may be contributing to the increased retention.

<table>
<thead>
<tr>
<th>Term</th>
<th>% of students who submitted assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2020</td>
<td>71%</td>
</tr>
<tr>
<td>Fall 2019</td>
<td>65%</td>
</tr>
<tr>
<td>Fall 2019</td>
<td>61%</td>
</tr>
</tbody>
</table>

This offering of the course was close to the same as the previous semester with the addition of videos to transition to online and more in class activity. The addition of videos will be incorporated in the future regardless of the format of the course. Many students commented on how he/she benefited from watching videos (or portions) multiple times to better understand challenging topics.

Proposed Changes:

CSCI 135 will be transitioning into CSCI 151. CSCI 151 will have the prerequisite of CSCI 100. Some of the content and teaching methods used in this course will be used in CSCI 151. CSCI 151 will also use the textbook: Introduction to Programming in Python – An Interdisciplinary Approach by Sedgewick, Wayne, Dondero. CSCI 151 will have more advanced learning outcomes:

1. Use/define functions: arguments, scope, side effects, return values
2. Create a client that uses libraries, APIs and Implementations
3. Problem solve using recursion, mathematical induction and dynamic programming
4. Create and use novel data types: classes, access modifiers, instance variables, constructors, instance methods, mutability
5. Develop software to optimize performance: scientific method, log-log plots, mathematical notation, order of growth classifications
6. Implement search and sort: binary search, insertion sort, merge sort, analysis of performance

Proposed Changes Requiring Approval of Undergraduate Studies Committee:

none

Grade Distribution:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>F</th>
<th>W</th>
<th>I</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>19</td>
<td>13</td>
<td>6</td>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>46</td>
</tr>
</tbody>
</table>
Faculty Course Assessment Report

Department of Computer Science
University of Montana, Missoula

CSCI 361
Computer Architecture (3 credits)

Spring 2020
Jesse Johnson


Prerequisites: CS136 – Fundamentals of Computer Science II

Income Assessment:

The students taking this course are mostly second year students, but a good number of third and fourth year students take it as well. The students have been exposed to structured and object oriented programming, requirements analysis, and the rudiments of algorithmic analysis. At this point, their exposure to hardware and the layers of abstraction that rest upon it is very limited.

Modifications made versus previous offering:
This course was previously assessed in 2017. The structure and content of the course have been relatively stable and changes from year to year have been primarily in terms of the in-class assessments.

The bottom up approach, expressed by the text’s sub-title “NAND to Tetris”, emphasizes that from the humblest of logic gates, one can construct, in its entirety, a modern computing platform. The route followed is 1) combinational logic and the ALU, 2) sequential logic and the CPU, 3) an assembler, 4) a virtual machine, 5) an object oriented language, and 6) a compiler for that language.

Some pedagogical features of the approach include:

- A bottom up approach, beginning with logic gates and concluding with compiler construction.
- Simulators are used instead of actual hardware.
- Assessment now takes the form of 6 programming assignments and three exams. With half the programming assignments on a simulator, and the other half using Python.
• Student’s present portion of the code they write to the class.
• Pre-testing of student understanding takes place during in-class quizzes, which are done in groups. The quiz is read and discussed prior to lecture and completed afterwards.
• Post-testing of students occurs on two, non-cumulative midterms and one cumulative final examination.

Unlike the two previous offerings of the course that took place since the last assessment, this offering stressed the pre and post-testing. As such, the outcomes based assessment will be structured around those results.

Course Outcomes Assessment:

In order to extend the baseline for assessment, the same outcomes were surveyed for Spring 2013, 2017, and 2020 offerings of the course. This is in spite of the significant changes in the way to course was offered in the last two. The results are tabulated below.

<table>
<thead>
<tr>
<th>Learning Goal Results 2013</th>
<th>2017 Modifications made to enhance learning</th>
<th>2020 Modifications made to enhance learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSCI 361 Computer Architecture Spring 2013</td>
<td>CSCI 361 Computer Architecture Spring 2017 Students appear to be able to do this, at least at the level of abstraction used in the previous course offering. Rather than an abstract mathematical construct like a polynomial, this year, students were assessed on the implementation of a stack in assembly language. Students were assessed with a programming assignment. Average score was 91.2%</td>
<td>CSCI 361 Computer Architecture Spring 2020 As was done in 2017, the stack was implemented in assembly. Questions over its implementation were stressed in pre- and post-tests. The post-test score is comparable to previous year’s results. The pre-test score is a new feature of the assessment. Both assessments were based on responses to several multiple choice questions. Pre-test average: 80% Post-test average: 93.6%</td>
</tr>
</tbody>
</table>

The student demonstrates the ability to write an efficient algorithm for an abstract mathematical operation, applying a relevant mathematical identity. In this case, students were asked to write an assembly-language procedure that evaluates a polynomial on a given point, and were prompted to use Horner’s identity.

Average score was 92.5%.
<table>
<thead>
<tr>
<th>Learning Goal Results 2013</th>
<th>2017 Modifications made to enhance learning</th>
<th>2020 Modifications made to enhance learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>The student demonstrates the ability to effectively implement recursion through use of a call stack alone. Students were given a high-level description of a recursive algorithm and asked to implement it in assembly language with appropriate use of the call stack.</td>
<td>To help students grasp this important concept, a more methodical approach to the call stack was taken in assembly, stressing each phase of the process, and what must be written to the stack in order to manage the calls. The Fibonacci sequence was used as a problem. Student understanding was assessed with a programming assignment.</td>
<td>As with the previous year, except this year the question was articulated on an examination instead of a programming assignment. As such, it was easier to focus questions on the stack frames and their composition, rather than investigate full set of recursive calls.</td>
</tr>
<tr>
<td>Average score was 80%.</td>
<td>Average score was 71.5%</td>
<td>Pre-test average = 81.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Post-test average = 84.4%</td>
</tr>
<tr>
<td>The student demonstrates the ability to translate high-level flow-control constructs into equivalent machine-level instructions. Students were asked to translate a given C algorithm (involving nested loops and array access) into an equivalent assembly program.</td>
<td>Changes in instruction were make the algorithms translated pseudocode, rather than C language. Student did multiplication and a device driver in assembly. This was assessed with a homework.</td>
<td>Again, the key difference was that these were moved to examination questions instead of programming assignments.</td>
</tr>
<tr>
<td>Average scores was 87.5%</td>
<td>Average score was 97.6%</td>
<td>Pre-test average = 45%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Post-test average = 42%</td>
</tr>
</tbody>
</table>

Reflection:
The pre- and post- test strategy reveals that students did not significantly improve between the time they were first exposed to a concept and the time it was examined over, generally a period of less than 8 weeks. It’s also true that the performance of students has not changes radically through the three assessments, except for the final assessment, which had to do with writing assembly for multiplication.

I attribute some of this to the change in what is assessed – test questions vs. programming assignments. I’ve long suspected that a lot of the code that is turned in on programming assignments is copied from the web, without a lot of depth of understanding. I prefer the more recent assessments that focus on test questions.

It is also true that reducing something as complex as a computer program to a multiple choice or short answer question can leave something out, and what students do know is not measured.

Where the pre- and post-test performance does not change significantly, I hypothesize that the way these are done, in groups, is leaving some students behind. That is measured in the post-tests, which are strictly individual based.

Spring of 2020 was interrupted by the Covid-19 virus. The transition to online learning impacted students in ways that are impossible to assess at this time.

Proposed Changes:

I will try and make groups that prepare the pre-test results more accountable to the individuals.

Proposed Changes Requiring Approval of Undergraduate Studies Committee:

Update the catalog description to reflect the changes made to the course.

Grade Distribution:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>12</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>37</td>
</tr>
</tbody>
</table>