Bloomsburg LOCK HAVEN MANSFIELD

COMMONWEALTH UNIVERSITY OF PENNSYLVANIA

Computer Science Assessment Plan 2024

Contents

Program Mission Statement

The Department of Mathematics, Computer Science, and Digital Forensics offers a Bachelor of Science degree in computer science. The curriculum is broadly based in core areas of computer science, with an emphasis on the design, analysis, and production of complex and reliable software. Graduates are prepared to advance in computing careers and lead in technical endeavors or pursue an advanced degree in computer science.

Program Educational Objectives

PEOs are broad statements describing the career and professional accomplishments that the computer science program prepares graduates to achieve.

Three to five years after graduation, our computer science alumni will:

1. be professionally employed in the computing field.
2. communicate and collaborate effectively in a team environment.
3. continue to grow professionally by adapting to new technologies and assuming leadership responsibilities.

Periodic Review and Revision

The Computer Science Curriculum Committee will review our mission statement and PEOs once every five years. Input from constituents will inform each review. This input will be obtained from advisory board members and alumni survey results.

Student Outcomes

We have ten SOs in six categories.

Category	SO: Student will…
Software Engineering	1. demonstrate strong programming skills in at least two object-oriented languages. 2. be able to write a significant application that efficiently utilizes a database for storage and retrieval. 3. be knowledgeable about software design processes and methodologies.
Operating Systems	4. have a strong understanding of operating system concepts.
Hardware	5. have a strong understanding of computer hardware concepts.
Problem Solving	6. be able to determine what abstract data type (ADT) should be used to solve a problem and what data structure would be used to efficiently implement an ADT. 7. be able to analyze the complexity of algorithms. 8. be able to solve programming problems.
Communication	9. demonstrate oral and written communication skills necessary to read, write, and speak effectively about concepts in computing.
Ethics	10. understand ethical and legal issues involving digital technology.

SOs Assessment Plan

Success in achieving the SOs is assessed through the administration of direct and indirect measures including the Major Field Test in Computer Science and various course embedded assessments. Indirect measures described after the following table help to assess our SOs as well as PEOs and our curriculum.

SO: Student will…	Direct assessment
1. Demonstrate strong programming skills in at least two object-oriented languages.	Course embedded assessments in CMSC 230 (Advanced Java) and CMSC 270 (Data Structures)
2. Be able to write a significant application that efficiently utilizes a database for storage and retrieval.	Course embedded assessment in CMSC 150 (Database Design)
3. Be knowledgeable about software design processes and methodologies.	ETS Major Field Test in Computer Science
4. Have a strong understanding of operating system concepts.	ETS Major Field Test in Computer Science
5. Have a strong understanding of computer hardware concepts.	ETS Major Field Test in Computer Science
6. Be able to determine what abstract data type (ADT) should be used to solve a problem and what data structure would be used to efficiently implement an ADT.	Course embedded assessment in CMSC 370 (Algorithms)
7. Be able to analyze the complexity of algorithms.	Course embedded assessment in CMSC 370 (Algorithms)
8. Be able to solve programming problems.	Problem Solving Assessment
9. Demonstrate oral and written communication skills necessary to read, write, and speak effectively about concepts in computing.	Course embedded assessments in CMSC 345 with presentation of final project and CMSC 480 with capstone project report.
10. Understand ethical and legal issues involving digital technology.	Course embedded assessment in CMSC 320. Students conduct an ethical analysis of a specified scenario involving the computing industry.

Indirect assessments: An exit survey of graduating seniors addresses all of our learning outcomes and allows us to determine students' perceptions of their education at the time of graduation. An alumni survey is sent to students three to five years after graduation, which helps us determine how they have continued their education and/or advanced in their careers. We also survey our advisory board members every three years for their thoughts and suggestions concerning our curriculum, PEOs, and SLOs.

Assessment tools

Assessment	Administered	Frequency	SLOs
Major Field Test	CMSC 480	Every spring	1-8
C++	CMSC 270	Once every 5 years	1
Java	CMSC 230	Once every 5 years	1
Database	CMSC 150	Once every 5 years	2
ADT and Runtime Analysis	CMSC 370	Once every 5 years	6-7
Problem Solving	CMSC 380	Once every 5 years	8
Communication	CMSC 345/480	Once every 5 years	9
Ethics	CMSC 320	Once every 5 years	10
Senior Exit Survey	Online	Every spring	1-10
Alumni Survey	Online	3-5 years after graduation	1-10 (and PEOs)
Advisory Board Survey	Online	Every 3-5 years	1-10 (and PEOs)

The Advisory Board Survey also solicits comments and suggestions about the CS curriculum in general.

Assessment Schedule

	S19	F19	S20	F20	S21	F21	S22	F22	S23	F23	S24	F24	S25
Review of mission statement											●		
Review of PEOs and SLOs											●		
Major Field Test	●		●		●						●		
C++ Assessment	●										●		
Java Assessment												☐	
Database Assessment												☐	
ADTs and Runtime Analysis										●			
Problem Solving													☐
Oral Presentation											●		
Written Assessment											●		
Ethics Assessment												☐	
Senior Exit Survey	●						●		●		●		
Alumni Survey													☐
Advisory Board Survey											●		

● = completed`
☐ = planned

Assessment Descriptions

1. Review of Mission Statement
2. Review of PEOs and SLOs

   The Computer Science Curriculum Committee meets at least once every five years to review our mission statement, PEOs, and SLOs, and to ensure that our curriculum remains aligned with these cornerstones. These assessments are informed by opinions solicited from external authorities, namely alumni who have advanced as software professionals, currently manage and hire developers, and/or entrepreneurs in the area of software development.

3. Major Field Test in Computer Science (MFTCS).

   This is our primary assessment tool. It is provided by ETS testing services (www.ets.org). This test is given to our graduating seniors every spring semester. It is required of students in CMSC 480, our capstone software engineering course. It allows us to compare our students to students at other universities and gives us a valuable external measurement with objective scoring and norm-referenced data. The test covers a broad spectrum of core computer science concepts and subject areas, providing objective criteria for assessing most of our SLOs:

   - Programming skills (SLO 1)
   - Software design processes and methodologies (SLO 3)
   - Operating systems (SLO 4)
   - Hardware (SLO 5)
   - Data structures and algorithms (SLOs 6-7)
   - Problem solving (SLO 8)

4. C++ Assessment

   This course-embedded assessment allows us to measure how well our students can design and implement software solutions in C++. It is administered in CMSC 270 (Data Structures in C++). The results allow us to assess SLO 1 (strong programming skills in at least two object-oriented languages) and SLO 8 (problem solving).

5. Java Assessment

   This course-embedded assessment allows us to measure how well our students can design and implement software solutions in Java. It is administered in CMSC 230 (Advanced Java). The results allow us to assess SLO 1 (strong programming skills in at least two object-oriented languages) and SLO 8 (problem solving).

6. Database Assessment

   This course-embedded assessment allows us to measure how well our students can design a database schema, implement SQL code, create reports, and solve a problem using a relational database. Students write code involving the use and/or creation of a database in two required courses, CMSC 150 (Database Design) and CMSC 230 (Advanced Java), and sometimes in CMSC 480 (Software Engineering). This assessment is administered in CMSC 150.

7. ADTs and Runtime Analysis

This assessment consists of selected questions from the final exam in CMSC 370 (Analysis of Algorithms). We collect data for individual questions (how many students answered them correctly) and individual students (how many questions they answered correctly).

8. Problem Solving

We refer to this assessment as a programming contest, not because the students are not competing against each other, but because the structure and administration of the assessment tool is similar to that of many high school and college programming contests. Students are given five programming problems of increasing difficulty to solve individually in three hours. Partial credit may be awarded for course grading purposes, but for assessment (which is transparent to the student) we consider each solution as correct or incorrect. We interpret the average number of problems solved as a measure of our students' abilities to solve programming problems—in fact, a measure of *general* problem-solving ability.

This assessment is given in CMSC 380 (Operating Systems), but the problems are independent of the course content. They are designed to require only general programming skills and do not rely on knowledge of standard libraries or special language features. Students may use the language of their choice since the correctness of a solution is judged by checking that it produces correct output for a range of hidden test cases.

9. Communication Skills

Oral skills are assessed with a report on a final project in CMSC 345 (Mobile Device Application Development). This course was taught face-to-face until 2022, after which we started offering it asynchronously online instead since it also serves as an elective for students in the Applied Computer Science program. Most of the students, however, are Computer Science majors, and only their presentations are used for assessment. The reports are now given in recorded video format, but the same criteria used for face-to-face presentations apply just as well in the new format.

Written communication skills are assessed using a written report due in CMSC 480 (Software Engineering).

10. Ethics Assessment

This course-embedded assessment is given in CMSC 320 (Computer Ethics, Social Impact, and Security). Students are given a software engineering scenario and asked to write an analysis of each actor's understanding of, and compliance with, professional responsibilities.

11. Senior Exit Survey

We developed an exit survey administered by the department office and taken every spring by graduating seniors. It allows students to state their perceptions of how well the program has satisfied learning outcomes and prepared them for graduate school or a position in the computing industry.

12. Alumni Survey

We remain in contact with many of our graduates as they advance in their careers. Sometimes they contact us with information about an internship or recent job posting at their company. Some stay in touch with one or more faculty members simply because they enjoyed their time here and the personal connections that

they made. Some of our former students have returned to speak in class about their professional experiences or to serve as a CS panelist for the CU's annual Career Day. We maintain a list of all such contacts and send email to them every three years with a link to a survey in order to measure how successfully we meet our Program Educational Objectives.

13. Advisory Board Survey

Our advisory board has been inactive since at least 2018. We are currently working to reconstitute it, and the following members have volunteered to serve.

- Len Kalechitz, class of 2001, founder and owner, Software Development Firm, LLC
- Colin Henry, class of 2004, Director of Software Engineering, Telly, Inc.
- Dan Polenik, class of 2014, Principal Software Engineer, Comcast, Inc.
- Brian Gorrie, class of 2018, Senior Software Engineer, Lockheed Martin, Inc.
- Brett Logan, class of 2018, Technical Team Lead, GitHub Expert Services

Rubrics and Surveys

The rubrics and surveys described in the previous section are included in successive pages following this one.

Contents:

C++ Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Pointers, operations on linked data structures, memory management	Little or no demonstrated understanding of how to perform dynamic memory allocation or manipulate pointers.	Missing or incorrect functions and/or obvious errors that may cause memory leaks.	Subtle errors that could cause memory leaks but all functions are implemented and operationally correct.	No potential memory leaks. Destructor, copy constructor, and assignment operator implemented correctly.	
STL iterators and sorting algorithms	STL is not used.	An STL vector and indexing is used instead of the required list class.	An STL list and an iterator are used with at most minor errors.	An STL list and iterator are used correctly and the list of objects is sorted properly.	
File I/O	Does not read any information from the input file.	Does not use C++ stream objects for file I/O, crashes, and/or does not read and store all the data in the file.	Uses C++ stream objects for file I/O, successfully reads and stores all the data in the file.	Uses C++ stream objects for file I/O, successfully reads and stores all the data in the file, using the most appropriate kind of loop, and closes the file.	
Operator overloading (and complexity requirement for operator+)	Little or no demonstrated understanding of how to overload operators and/or invoke them.	Significant gaps in knowledge of how to overload operators and/or invoke them. Operator+ does not meet complexity requirement.	Operator overloading is generally correct, but complexity requirement for operator+ is not met.	Required operators are correctly overloaded, and complexity requirement for operator+ is met.	
Templates	No attempt to implement a class template.	Major errors in class template, e.g., a member function is not templatized.	No major errors. Class template can be instantiated and is functional.	No errors. Complies with coding conventions.	
General OOP principles	Incorrect parameter and return value types, global variables or other details that subvert the idea of information hiding, incorrect use of const.	Interface lacks cohesion. No understanding of when/why to declare references and methods const. Member functions not focused on their particular responsibilities.	Public interface contains one or two member functions not related to the concept represented by the class. Member functions or references not consistently declared const when they should be.	Parameters and return values are declared with appropriate types. Const is used where appropriate. No global variables or other hacks to violate information hiding. Clear separation of public interface and private implementation. Cohesive public interface.	
Clarity	Significant deviations from coding standards throughout. Many parts of the code are undocumented, overly complex, and/or cannot be understood without judgment or guesswork.	Significant deviations from coding standards. The code is disorganized or poorly documented, and difficult to understand in places.	Code is generally easy to read, but in some cases insufficient documentation, inconsistent indentation, cluttered or overly complicated code, or other minor deviations from coding standards.	The code is professionally written: neatly organized, easy to read and understand, with correct indentation, reasonable choices for identifiers, and internal documentation to explain non-obvious details of the logic or its implementation.	
				TOTAL	

Student:

Evaluator:

Java Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Implementing Interfaces	No attempt to implement the Comparable interface	Incorrectly implemented the Comparable interface	The Comparable interface is implemented correctly in most instances and classes.	The Comparable interface is implemented correctly in all the appropriate classes.	
Object-Oriented Design	Difficult to follow design.	Some good design elements, but many design problems are evident.	Reasonable class design, but some design problems are evident.	Excellent class design throughout the entire project.	
Generic Class Design	No attempt to use generic types.	Generic types are used, but there are many problems with their specifications and implementations.	Generic types are used correctly in most cases.	Generic types are used correctly in all cases.	
Coding Style	Code is difficult to read and understand due in part to major violations of standards for coding good style.	Code is generally readable but violates many standards for good coding style.	Complies with most standards for good coding style.	Consistently complies with all standards for good coding style.	
JavaDoc	Minimal documentation, or most methods are not correctly documented.	Many methods are not correctly documented.	Most methods are documented correctly and completely.	Each method and class has an appropriate doc comment with block tags as needed.	
Code	Code does not execute.	Code executes, but many implemented methods do not perform correctly.	Most implemented methods perform correctly.	The program works correctly and all methods are implemented correctly.	
Problem Solution	Many program requirements are not completed.	Most requirements are completed, but few are correct.	Most requirements are completed correctly.	All requirements are completed correctly and the program is user friendly.	
				TOTAL	

Student:

Evaluator:

Database Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Database Design	Table structure is difficult to follow. Not all required information is represented.	All required information is represented, but the table structure is poorly designed.	Table structure is appropriate and all required information is represented.	Table structure is well designed and all required information is represented. Tables have a primary key.	
Table Creation Statements	SQL code to create the tables is mostly incorrect or poorly designed.	Some SQL code to create the tables is correct, but many items are incorrect or poorly designed.	Most SQL code to create the tables is correct, but one or two columns are of the wrong type.	All SQL code to create the tables is correct.	
Insert Statements	SQL code to insert data into the tables is mostly incorrect or poorly designed.	Some SQL code to insert data into the tables is correct, but many items are incorrect or poorly designed.	Most SQL code to insert data into the tables is correct, but one or two columns are of the wrong type.	All SQL code to insert data into the tables is correct.	
Other SQL Code	Most code does not execute correctly.	Some of the SQL statements execute correctly, but many methods do not perform correctly.	Most implemented methods perform correctly.	The entire program is correct. All methods are implemented correctly.	
Reports	Most reports are poorly designed and unsatisfactory.	Many reports are poorly designed and unsatisfactory.	Virtually all reports are well designed and implemented.	All reports are well designed and implemented.	
Problem Solution	Many solution requirements are not completed.	Most requirements are completed.	Solution is well done with only a few minor issues.	All requirements are completed. Project is easy to use and understand.	
				TOTAL	

Student:

Evaluator:

ADT and Runtime Analysis Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Analysis of Iterative Algorithms	0 - 35% correct	36 - 60% correct	61 - 85% correct	86 - 100% correct	
Analysis of Recursive Algorithms					
Application of Critical Thinking to Choosing Appropriate ADTs, Data Structures, and Algorithms					
				TOTAL	

Evaluator:

Student:

Oral Communication Assessment Rubric

Speaker: Evaluator:

Topic: Date:

Evaluation scale:

- 4 = Excellent
- 3 = Good
- 2 = Marginal
- 1 = Unsatisfactory

Presentation Style

	Score	Weight	Total
Personal appearance was appropriate.		1	
Maintained eye contact with audience.		1	
Used audience-appropriate vocabulary.		1	
Paced the presentation appropriately.		1	
Used engaging vocalizations.		2	
Maintained audience interest.		2	
Spoke clearly, confidently, and with sufficient volume.		2	
Smooth transitions between topics with limited use of “ums” and other filler words.		2	

	Score	Weight	Total
Presentation includes introduction, body, and conclusion.		3	
Content is logically organized.		3	
Visual aids and/or presentation materials enhance the presentation.		3	
Demonstrates subject knowledge and responds effectively to questions.		4	

Content

Weighted Total	/100

Evaluator:

Date:

Written Communication Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Grammar and spelling	Many sentences have grammar or spelling errors.	Most paragraphs have a grammar or spelling error.	Most paragraphs have no grammar or spelling errors.	The entire work has at most a couple of grammar or spelling errors.	
Sentence structure	Run on and awkward sentences occur in most paragraphs.	Some run on and awkward sentences are present. Sentence structure varies little.	Very few run-on and awkward sentences are present. Sentence structure is usually varied appropriately.	No run on or awkward sentences. Sentence structure is varied appropriately.	
Paragraph structure	Most paragraphs are incoherent.	Some paragraphs are structured appropriately.	Most paragraphs are structured and obviously coherent.	Every paragraph is begun, developed and concluded appropriately.	
Composition structure	Ideas appear haphazardly or incompletely. Relationships among ideas are not evident.	Ideas are present but often unrelated. Main points are not evident. Pacing uneven.	Main points are evident and usually related in a logical fashion. Introduction and conclusion are present.	The subject is introduced and main points developed. Conclusions follow from main points.	
				TOTAL	

Student:

Evaluator:

Date:

Computer Ethics Assessment Rubric

	Unsatisfactory 1	Marginal 2	Good 3	Excellent 4	Score
Ethical Arguments	Ethical arguments do not match the ethical system.	Ethical arguments are appropriate for the ethical system, however the reasoning skills demonstrated are weak or incomplete.	Almost all ethical arguments demonstrate strong reasoning skills in the ethical system. Arguments are mostly complete.	Ethical arguments demonstrate strong reasoning skills in the ethical system. All arguments are complete and concise.	
Primary actors are identified in the professional ethics scenarios.	Little or no identification of primary actors is completed.	Some primary actors are correctly identified.	Most primary actors are correctly identified.	All primary actors are correctly identified.	
Professional responsibilities are identified in professional ethics scenarios.	Few or no professional responsibilities are identified.	Some professional responsibilities are identified, but many are missed or too many actors are listed.	Most professional responsibilities are correctly identified, with few superfluous responsibilities listed.	All professional responsibilities are correctly identified, without superfluous responsibilities listed.	
Ethical resolution of the scenario is identified.	Little or no judgment has been made as to ethical resolution of the scenario	Some judgments are made as to as to the correct ethical resolution of the scenario. Little or no justification for judgments is present.	Mostly correct judgments are made as to as to the correct ethical resolution of the scenario. Most judgments are supported by valid reasoning.	Completely correct judgments are made as to as to the correct ethical resolution of the scenario. All judgments are supported by valid reasoning.	
				TOTAL	

Student:

Evaluator:

Date:

Senior Exit Survey

Name:

Year of graduation:

Permanent email address (optional):

---

1. Are you currently seeking employment?

2. Have you been offered a full-time position but not yet accepted? If so, could you briefly describe the options you are considering?

3. Have you accepted a full-time position? If yes:

- What is the job title?
- Who is the employer?
- Please share with us any experiences as a CS major that were especially important in your being hired. This could be a particular course (or courses) that you took, particular skills developed or concepts covered in the curriculum, an internship or other employement experience, or anything else that you did or learned as a CS major that made you a good candidate for the position.

4. Have you been accepted to a graduate school program? If so, will you be pursuing a Masters degree or a Doctorate? Please tell us name of the program and the school.

5. Having completed our CS program, how prepared do you feel for your next step?

- Very prepared
- Reasonably well prepared
- Somewhat prepared
- Poorly prepared

If you feel less than reasonably well prepared for your next step, please tell us why.

6. Please check the box to indicate how prepared you feel in the following areas.

	Poor	Satisfactory	Good	Excellent
Java programming skills				
C++ programming skills				
Object-oriented programming in general				
Ability to write a significant database application				
Understanding of computer hardware concepts				
Understanding of data structures and algorithms				
Knowledge of operating system concepts				
General problem-solving skills				
Proficiency in oral and written communication				
Understanding of ethical issues related to computing				
Knowledge of software design processes and methodologies				

7. Describe what you liked most about the CS program?

8. Describe what you liked least about the CS program?

9. Do you have any suggestions for how the program could better serve its students?

10. Do you have any additional comments about the CS program at Bloomsburg or your experiences as a CS major?

Alumni Survey

Name:
Date:
Year of graduation:

The first seven questions assume that you are currently employed in the computing field. If this is not the case, please proceed to Question 8.

1. Who is your current employer?

2. What is your current job title?

3. Please summarize your current professional responsibilities. We are especially interested in hearing about any leadership roles that you may hold.

4. Does your position require the ability to communicate and collaborate effectively in a team environment? If so, how well did your experience as a computer science major help prepare you for this aspect of your career?

5. From your current perspective as a computing professional, how would you rate your overall level of preparation for a career in computing at the time of your graduation?

- 1 = Poor
- 2 = Adequate
- 3 = Good
- 4 = Excellent

6. If you answered 1 or 2 for the previous question, please explain.

7. We would like to know how well the computer science major prepared you in the areas listed below for entry-level responsibilities in your position. Select N/A if you cannot judge or an area is not relevant to your position.

	N/A	Poor	Adequate	Good	Excellent
Java programming					
C++ programming					
Database design and implementation					
Data structures and algorithms					
Operating systems					
Software engineering					
Problem-solving					
Oral communication					
Written communication					
Leadership skills					
Ability to adapt to new technologies					
Ability to work in a team environment					

8. If you are currently in graduate school or have completed a graduate degree, please tell us about it:

Degree:

Program:

School:

Date (or expected date) of completion:

9. We welcome any comments you may have about the computer science program at CU - Bloomsburg and/or suggestions for the improvement of the program

Advisory Board Survey

Name:

Title/Position:

Company/Employer:

One of the requirements for our ABET-accredited CS program is to periodically assess our Program Educational Objectives (PEOs) and Student Outcomes (SOs). PEOs are broad statements describing what graduates are expected to attain within a few years after graduation. They are based on the needs of the program's constituencies. Most of our graduates enter into some form of software development, making the software industry our primary constituency. SOs describe what students are expected to know and be able to do at the time of graduation.

Please review our PEOs and SOs here, and then answer the following two questions.

1. From your perspective as a computing professional, are our PEOs clear, sufficient, and appropriate to meet the needs of our constituents? Is there anything you would change or add?

2. How well do you think our Student Outcomes support the PEOs? Considering recent trends in software development and other areas of computing, would you recommend any changes or additions to our SOs?

Please take a look at our Degree Checklist here in order to answer the next question.

3. Do you think the required and elective courses align with our SOs and with the general needs of the software industry? Are there areas within the curriculum that you believe deserve extra emphasis, areas not optimally covered for today's needs, or areas that receive more emphasis than is justified by today's needs?

4. Finally, we welcome any general comments and/or suggestions you may have for the improvement of our computer science program.

Thank you for your time.