Bloomsburg University

Computer Science Assessment Report

2016

Introduction

Bloomsburg University of Pennsylvania (BU) is one of fourteen universities in the Pennsylvania State System of Higher Education. BU has approximately 9,000 undergraduate students and 1,000 graduate students. The Department of Mathematics, Computer Science and Statistics is in the College of Science and Technology (CoST) and services undergraduate students with four undergraduate degree programs and five minors. The department houses twenty-five tenured/tenure-track faculty members, all of whom hold a doctorate degree in their respective fields. The Computer Science program has a steady state of about 125 majors and is supported by seven faculty members.

Computer Science Program Mission Statement

The Department of Mathematics, Computer Science and Statistics 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 will be prepared to advance in computing careers and lead in technical endeavors, or to pursue an advanced degree in computer science.

Computer Science Program Educational Objectives

Program educational objectives are broad statements that describe the career and professional accomplishments that the computer science program is preparing graduates to achieve.

Three to five years after graduation, our computer science alumni will be pursuing an advanced degree or they 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 Process

The Computer Science Curriculum Committee reviews our mission statement and program educational objectives once every five years.

Computer Science Program Student Learning Outcomes

The Computer Science Program has ten program learning outcomes listed under six categories.

- **Software Engineering**
    1. Students will demonstrate strong programming skills involving at least two object-–oriented languages.
    2. Students will be able to write a significant application that efficiently utilizes a database for data storage and retrieval.
    3. Students will be knowledgeable about software design processes and methodologies.

- **Operating Systems**
    4. Students will have a strong understanding of operating system concepts.

- **Hardware**
    5. Students will have a strong understanding of computer hardware concepts.

- **Problem Solving**
    6. Students will be able to determine what Abstract Data Type (ADT) should be used to solve a problem and what data structure should be used to efficiently implement an ADT.
    7. Students will be able to analyze the complexity of algorithms.
    8. Students will be able to solve programming problems.

- **Communication**
    9. Students will demonstrate oral and written communication skills necessary to read, write, and speak effectively about concepts in computing.

- **Ethics**
    10. Students will understand ethical and legal issues involving digital technology.

Periodic Review and Revision Process

The Computer Science Curriculum Committee reviews our Student Learning Outcomes once every five years

Summary of All Assessment Methods

Assessment Method	Administered	Frequency	Reviewed	Relevant Outcomes
Major Field Test in Computer Science Assessment	During course COMPSCI 480	Every spring semester	Following fall semester	1, 3, 4, 5, 6, 7, and 8
C++ Assessment	During course COMPSCI 255	Once every 3 years.	Following semester	1
Java Assessment	During course COMPSCI 221	Once every 3 years.	Following semester	1 and 2
ADT and Runtime Analysis Assessment	During course COMPSCI 355	Once every 3 years.	Following semester	6 and 7
Programming Problem Solving Assessment	During course COMPSCI 386	Once every 3 years.	Following semester	8
Communication Skills Assessment	During course COMPSCI 480	Once every 3 years.	Following semester	9
Ethics Assessment	During Course COMPSCI 120	Once every 3 years.	Following spring semester	10
Graduating Senior Survey Assessment	End of semester	Every Year	Following semester	All
Graduate Assessment	Online Survey	Once every 3 years.	Following semester	All
Employer Assessment	Online Survey	Once every 3 years.	Following semester	All

Advisory Board Meeting	Career Day	Once every 3 years.	Same Semester	All

Assessment Activities and Results

All assessment activities are planned, scheduled, administered, reviewed and discussed by the Computer Science Curriculum Committee. In this section, we list each assessment activity, show the results of the assessments conducted since we started our assessment activities in 2007. We include comments about our students' performance and any changes we are making based on the review of this data.

Review of Mission Statement and Program Educational Objectives (PEO)

This activity was last completed during the fall 2013 semester and is scheduled to be completed again during the fall 2018 semester. The Computer Science Curriculum Committee (CSCC) utilized Advisory Board member comments, alumni survey results and comments, employer survey results and comments, graduating senior survey results, comments from high school computer science educators during our annual programming contest along with a review of the PEO of similar programs at other universities to review our program's educational objectives.

This comprehensive review resulted in one minor editorial change, but the CSCC determined that our mission statement and PEO are appropriate, accurate, and serve the needs of our program and its constituents.

Review of Student Learning Outcomes

This activity was last completed during the fall 2013 semester and is scheduled to be completed again during the fall 2018 semester. The CSCC utilized Advisory Board member comments, alumni survey results and comments, employer survey results and comments, graduating senior survey results, comments from high school computer science educators during our annual programming contest along with a review of our curriculum to review our program's Student Learning Outcomes (SLO)

This comprehensive review resulted in several editorial changes, but the CSCC determined that our SLO are still appropriate, accurate, and serve the needs of our program and its constituents.

Advisory Board Meeting

Our last meeting was during the fall 2013 semester and our next meeting is scheduled for Career Day in October 2017. We are planning to hold advisory board meetings to coincide with our Career Day activities which have now grown into a College of Science and Technology event.

The main focus of our last meeting and the meeting before that was held during the spring 2013 semester was a review of our Computer Science Program's Mission Statement, Program Educational Objectives and Student Learning Outcomes. We also discussed our curriculum and what they see as the future needs of employers in computer related fields.

Advisory Board Members along with their year of graduation, employer and current position

Len Kalechitz 2001, Solution Development Firm, LLC, Computer Scientist
Scott McCarty 1998, OPTiMO, Director of Information Technology
James Campbell 1998, Penn State University, Senior Unix Consultant
Matthew Quinn 2002, The Pennsylvania State University, Applied Research Laboratory
Mike Trelease 1998 and 2006, Geisinger Health System, Program Director
Dan Miller, 2005, Hershey Corporation, Application Development Director
Mitch Parker, 1997, Temple University Health System, Chief Information Security Officer

Major Field Test in Computer Science (MFT)

This activity was last administered during the spring 2016 semester to 16 graduating seniors and will be completed again during the spring 2017 semester. We currently ask all our graduating seniors to complete this assessment during our senior capstone course, COMPSCI 480, Object-–Oriented Software Engineering. Their performance is used to partially determine their final grade in this class. We have found that students take the MFT seriously when it is used as part of their grade in a course. Our students have performed well in this assessment activity. Our department has ranked from the 68th to 98th percentile since we first started administering this exam during the 2006 academic year.

Over these ten academic years, our average student scored a 156 out of 200 possible points, which places us as a department in the 79th percentile of all participating programs. We originally expected our students to perform in the 50th percentile range, so we are pleased with the performance of our students in this assessment activity.

The table below shows how our students performed collectively in the three major content areas of the MFT during the 2016 academic year. This year our departmental average was 156, which placed us in the 79th percentile out of 132 schools administering this assessment activity. This result exceeded our original expectations and we are proud of how well our students are performing.

2015-–2016 MFT Results by Content Area

Assessment Indicator Number	Assessment Indicator Title	Our Mean Percent Correct	Department Percentile
1	Programming and Software Engineering	67	90
2	Discrete Structures and Algorithms	47	59
3	Systems: Architecture/Operating Systems/Networking/Database	50	76

We have placed additional results from the MFT assessment activity in the appendix.

C++ Assessment

This activity was last administered during the spring 2016 semester. We assess the students on their abilities in seven major areas of using C++ to solve a programming problem. The students are scored on a 4-point scale with 4 being excellent and 1 being unsatisfactory. The rubric is contained in the appendix. Our goal is to average a score of 3 or better in each category.

C++ Assessment Results

	Pointers	STL	File/IO	Operator Overloading	Templates	General OOP	Clarity
2016	2.67	2.95	2.67	2.62	3.19	2.76	2.76
2013	3.25	3.29	2.83	3.67	3.58	3.33	3.25
2011	3.04	3.39	3.07	3.51	3.72	3.21	2.99
2010	3.14	3.72	3.25	3.58	3.72	3.36	3.03
2007	3.09	2.91	2.55	3.45	3.55	3.64	2.73

We determined these results are acceptable, but plan to place more emphasis on Pointers, File I/O, and Operator Overloading in future semesters.

Java Assessment

This activity was last administered during the fall 2016 semester. We modified our assessment assignment and criteria after the 2013 assessment evaluation. We determined it would be better to separate the database activities out of the Java assessment and concentrate on more direct object-oriented criteria. Please see the discussion at the bottom of this page for more information on why we modified the assessment vehicle for this student learning outcome. The students are scored on a 4-point scale with 4 being excellent and 1 being unsatisfactory. The rubrics are contained in the appendix. Our goal is to average a score of 3 or better in each category.

Java Assessment Results
2016

Year	Interface Design	Object-Oriented Design	Generic Class Design	Java Coding Style	Java Documentation	Code	Problem Solution
2016	2.76	3.48	3.14	3.19	2.86	2.81	2.71

Overall, we find the results from 2016 acceptable. We will look to improve our coverage of interface design to help students improve the overall quality of their solutions.

Java Assessment Results
2007 to 2013

Year	Database Design	Object--Oriented Design	GUI Design	Java Coding Style	Java Documentation	Code	Problem Solution
2013	2.96	2.67	2.80	3.39	3.13	3.39	3.17
2009	2.18	3.27	2.84	3.36	3.05	2.91	2.82
2007	3.00	2.90	2.90	3.10	2.50	2.75	2.65

In 2013 we determined that the programming assignment used for this activity does not allow us to adequately determine how well our students understand object--oriented design because the project was a complete web application with Java Server Pages and Servlets. We developed a better assessment instrument and rubric for object--oriented design to properly assess our students starting with the 2016 assessment. We removed the database design component out of this assessment vehicle and move it into its own assessment activity. We determined that additional assessment categories for database design and database use in solving a programming problem was appropriate. Overall, we are pleased with the results in five of the seven categories. Curriculum changes

completed after the 2007 assessment were implemented to address the low assessment scores in several categories of Java programming (documentation, better coding skills, and problem solutions) and have resulted in a better understanding of these concepts by our students.

ADT and Runtime Analysis

This activity was last administered during the fall 2013 semester. We assess the students on their abilities to analyze the runtime behavior of iterative algorithms, recursive algorithms and their choice of Abstract Data Types and Data Structures to solve computationally difficult programming problems. The students are scored on a 4-point scale with 4 being excellent and 1 being unsatisfactory. Our goal is to average a score of 3 or better when analyzing iterative algorithms and the choice of ADTs and data structures. We would like the students to average a 2.5 or better when analyzing recursive algorithms. We find that analyzing recursive algorithms is difficult for undergraduates and it is not a major area of stress in courses. The rubric is contained in the appendix.

ADT and Runtime Analysis Results

	2013	2011	2010	2009
Analysis of Iterative Algorithms	3.43	3.32	1.97	3.38
Analysis of Recursive Algorithms	2.71	3.05	3.18	2.23
Choice of ADT, D/S, Algorithms	3.21	3.05	2.75	3.31

We are pleased with the results of this assessment activity.

Programming Problem Solving Assessment

This activity was last administered during the spring 2016 semester. We assess students on their ability to solve five different programming problems in a three-–hour time limit. It is essentially a programming contest with teams of size one. We expect our students to solve three of the five problems on average. The problems range in difficulty from easy to difficult. We believe that all students should solve the first problem. Most students should solve the second problem. The average to good students should finish the first three problems. Very good students will be able to solve the first 4 problems in the time allotted, and excellent students can solve all 5. We expect that solving all 5 problems will be a rare event. That has turned out to be true. In the five years we completed this assessment activity, only three students have solved all five problems.

Programming Problem Solving Assessment Results

Year	Average Number of Problems Solved	Percentage of students solving exactly 1 to 5 problems respectively*
2016	2.8	8%, 13%, 56%, 0%, 8%
2013	3.4	6%, 6%, 35%, 47%, 6%
2012	3.0	8%, 46%, 8%, 31%, 8%
2011	2.6	33%, 17%, 8%, 42%, 0%
2010	3.1	21%, 0%, 21%, 57%, 0%

*Rounding errors may keep these numbers from totaling 100%

Overall; we are pleased with the results of this assessment activity.

Oral Presentation Assessment

This activity was last administered during the Fall 2015 semester. We created our own assessment rubric that considers both how the speaker performs in presentation style and the content of the information being delivered. We have eight aspects of giving a presentation that we determine to be important in the presentation style portion of the assessment and four areas that we deem important in the content category. Students are evaluated over these twelve areas in a range of 1 to 4 to be consistent with all our assessment rubrics. We use a weighted total to yield an overall score out of 100 points to place greater weight on items that we feel are slightly more important than other characteristics of presenting well. The oral presentation assessment rubric is in the appendix. Multiple faculty members rate each student. Each student is given the average score of all the faculty members viewing the presentation and these scores are averaged for the academic year. The table below shows how the students did in each category. We expect our students to average three or above in each category and have a weighted total of 75 or above in this assessment activity.

Average Student Scores in our Oral Presentation Assessment

Year	Number of Students	Personal Appearance	Speaks Clearly	Smooth Transitions	Confident Speaker	Avoids Distractions	Appropriate Vocabulary
2015	14	2.79	3.04	2.89	2.96	3.29	3.43
2012	15	3.24	3.3	3.26	2.92	3.29	3.63
2011	10	3.19	3.33	3.2	3.02	3.3	3.78
2010	8	3.55	3.43	3.35	3.29	3.38	3.61
2009	10	3.29	3.10	3.00	3.05	3.06	3.35
2008	12	3.28	3.54	3.15	3.29	3.37	3.62
2007	5	3.12	3.32	2.88	3.12	3.24	3.64

Year	Eye Contact	Maintains Interest	Introduction, Body, and Conclusion	Logically Organized	Visual Aids	Subject Knowledge	Weighted Points
2015	3.25	3.14	3.32	3.54	2.96	3.36	80.32
2012	3.24	3.15	3.39	3.62	3.34	3.47	84.44
2011	3.02	3.13	3.64	3.82	3.52	3.6	86.93
2010	3.25	3.09	3.45	3.61	3.35	3.65	86.56
2009	2.95	3.08	3.60	3.59	3.39	3.57	83.40
2008	3.33	3.24	3.60	3.61	3.43	3.55	86.62
2007	3.2	3.32	3.4	3.44	3.36	3.4	83.12

We are pleased with our students' performance when giving presentations. This is likely the result of our general education program at Bloomsburg University that stresses communication skills, the overall quality of a student that is needed to complete a computer science major and the fact that our students give short presentations in several computer science classes. We require our students to take Foundations of Writing, Technical Writing and Public Speaking classes as part of their general education requirements along with three laboratory sciences. Students are required to complete weekly lab reports in the lab science courses.

Written Assessment

This activity was last administered during the Spring 2015 semester. We created our own assessment rubric that considers four components of a student's writing ability. Students are evaluated over these four areas in a range of 1 to 4 to be consistent with all our other assessment rubrics. The writing assessment rubric is in the appendix. One computer science faculty member and a technical writing faculty member evaluate each written document and the results are averaged for each student. The table below contains the average of all the writing samples for the academic year. The writing samples are three to five page papers written by senior computer science students. We expect our students to average three or above in each category

Writing Assessment Results

	Number of students	Grammar & Spelling	Sentence Structure	Paragraph Structure	Composition Structure	Average
2015	19	2.29	2.36	2.78	2.72	2.54
2012	14	3.00	3.07	3.36	3.63	3.26
2011	8	3.28	3.03	3.53	3.53	3.34
2010	8	3.38	3.16	3.47	3.53	3.38
2009	9	3.01	2.99	3.11	3.26	3.03

Overall, we are pleased with our student's writing ability. The most recent evaluation has given us some cause for concern. We will continue to monitor this assessment to see if a downward trend continues Similar to our student's public speaking abilities, the usually high level of skills are likely the result of several factors. Some of these factors include an appropriate general education program that stresses communication skills along with multiple classes that provide the students with opportunities to express their ideas in written form along with feedback on their work. The fact that we have a technical writing faculty member in our department is also a plus.

Ethics Assessment

This activity was last administered during the fall 2013 semester. Freshmen computer science students were assessed on their ability to determine which imperatives from the ACM Code or Ethics apply to sample scenarios. This assignment was given at the end of a 1-- – credit course during the student's first semester on campus. They are also given similar final exam questions. The rubric is in the appendix and we would like our students to score a three or better for each scenario. The results of this assessment are shown below.

Ethics Assessment Results

Year	Students	Scenario 1	Scenario 2	Scenario 3
2013	43	2.35	2.44	2.72
2009	25	2.88	3.44	2.68

Our freshmen did not meet our expectations in this assessment. Part of the issue is that they are freshmen and this is a one credit class that meets 50 minutes a week for 14 weeks. We are also trying to cover too much material in this one-–credit freshmen class. The students are assessed during the course, so they do not have any additional time to reflect on the knowledge or apply it in other classes. They also do not have any real experience with ethical situations regarding the use of technology. We have removed this one credit freshmen class from our curriculum and have replaced it with a three-–credit junior level course. This will give us additional class time to cover computer ethics and the impact of computers on society. The students will also have a richer set of experiences to draw from during class discussions. The fall 2014 freshmen class will be the first cohort of students with this new requirement.

Senior Exit Survey

This survey was last given during the spring 2016 semester. We ask our graduating seniors to complete a survey as part of their graduation process. This survey is in the appendix. We ask our students for their opinions about their experience at BU and our program. The students state their thoughts on how well our program prepared them overall as a computer scientist and how well prepared they feel in particular areas that related to our student learning outcomes. The students are given four choices and these responses are translated into a 1 to 4 scale with four being excellently prepared and one being poorly prepared. We strive for students to rate their abilities at a three or above for each of the categories. A tabulation of the student responses are shown in the two tables below. Students also provide us with many written comments that we discuss during Computer Science Curriculum Committee meetings.

Year	OOP	Java	C++	Database Applications	Software Design	OS Concepts	Hardware	Problem Solving
2015 -- 2016	3.52	3.52	2.91	2.39	2.48	2.83	2.26	2.91
2014 -- 2015	3.95	3.85	3.05	2.80	3.15	2.8	2.45	3.75
2013 - 2014	3.79	3.93	3.14	2.86	3.29	2.86	2.64	3.64
2012 – 2013	3.43	3.57	2.5	2.64	2.93	3.38	2.29	3.43
2011 – 2012	3.60	3.60	2.70	2.70	3.10	2.56	2.50	3.50
2010 – 2011	3.77	3.69	3.00	2.85	3.08	3.08	2.77	3.69
2009 – 2010	3.55	3.45	2.73	2.73	3.18	2.82	2.73	3.18

Year	Algorithms	Oral & Written Communication	Ethical and Legal Aspects	Overall Prepared	Students Completing Survey
2015 -- 2016	2.78	3.13	2.87	3.08	23
2014 -- 2015	3.05	2.95	2.90	3.25	20
2013 - 2014	3.14	3.36	3.29	3.43	14
2012 – 2013	3.07	2.86	2.85	3.21	14
2011 – 2012	3.10	3.10	2.80	3.20	10
2010 – 2011	3.46	3.31	3.15	2.92	13
2009 – 2010	2.82	2.64	3.00	3.09	11

Overall, the students are positive about their experiences in our program. Our curriculum is weighted towards software development and programming skills. We only have one required hardware course and one course on operating systems concepts. We find the student perceptions match our own view on our curriculum and its strengths and weaknesses.

Alumni Survey

This activity was last administered during the fall 2013 semester. We emailed alumni who graduated in 2008 to 2010 to complete an anonymous survey. Twenty-–seven alumni responded and their responses are summarized in the table below. They were asked to rate themselves and their responses were converted into numbers with 4 being excellently prepared or having excellent skills and 1 being poorly prepared or having poor skills. We would like to see our alumni rate themselves at the equivalent of a 3 (good) or higher in each category. An exported version of the Qualtrics survey they completed is contained in the appendix. The online version is formatted differently, but this version will show the reader how questions were phrased.

Alumni Survey Results
27 alumni responded

2013 Survey	**OOP**	**Java**	**C++**	**Database Applications**	**Software Design**	**OS Concepts**	**Hardware**
Average Rating	3.7	3.5	2.8	2.9	3.3	2.6	2.7
	Algorithms	**Problem Solving**	**Oral & Written Communication**	**Overall Prepared**	**Team Player**	**Adapt to New Technologies**	**Leadership Skills**
Average Rating	3	3.8	3.3	3.0	3.7	3.7	3.4

The survey results are basically what we expected. It seems many of our graduates are not in application areas that utilize the C++ language and the graduates who responded feel their skills in C++ are below their skills in Java.

Employer Survey

This survey was activated during the fall 2013 semester. Nine supervisors responded indicating that they supervised a total of 38 graduates of our program. It makes sense that employers who have multiple individuals from BU would be more inclined to respond to our survey and report their observations. This survey is included in the appendix. We asked the supervisors to rate our graduates over thirteen areas that relate to our Program Educational Objectives and our Student Learning Outcomes. The rankings were converted to a four point scale with excellent skills being a 4 and poor skills being a 1. We expect our graduates to be at a level equivalent to a 3 (good) or higher three to five years after graduation. The following table shows the average of the employers' responses in the top column and the weighted average that multiplies each supervisor's rating by the number of graduates supervised.

Employer Survey Results
Nine Supervisors reporting on 38 graduates

2013 Survey	**OOP**	**Java**	**C++**	**Database Applications**	**Software Design**	**OS Concepts**	**Hardware**
Average Rating	3.6	3.5	3	3.3	3.3	3.3	3
Weighted Average	3.9	3.8	2.9	3.6	3.0	3.4	3.3
	Algorithms	**Problem Solving**	**Oral Communication**	**Written Communication**	**Team Player**	**Adapt to New Technologies**	**Leadership Skills**
Average Rating	3.3	3.4	2.9	3.1	3.1	3.6	2.4
Weighted Average	3.7	3.8	2.9	3.7	3.8	3.7	2.7

For the most part, the employer responses are what we expected, but it was slightly more difficult getting them to respond than we anticipated. We are surprised that the Oral Communication rating is below a 3.

Summary of Student Learning Outcomes (SLO) Assessments

Student Learning Outcome	Assessment Comments
SLO 1: Students will demonstrate strong programming skills involving at least two object-–oriented languages.	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. Java Assessment C++ Assessment MFT Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 2: Students will be able to write a significant application that efficiently utilizes a database for data storage and retrieval.	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. Java Assessment MFT Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 3: Students will be knowledgeable about software design processes and methodologies	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. Java Assessment C++ Assessment MFT Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 4: Students will have a strong understanding of operating system concepts.	Satisfied. Evidence to support the satisfactory

	completion of this learning outcome by our students is contained in the following assessments. MFT Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 5: Students will have a strong understanding of computer hardware concepts	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. MFT Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 6: Students will be able to determine what Abstract Data Type (ADT) should be used to solve a problem and what data structure should be used to efficiently implement an ADT. Students will be able to solve programming problems.	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. MFT Assessment Program Problem Solving Assessment ADT and Runtime Analysis Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 7: Students will be able to analyze the complexity of algorithms	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. MFT Assessment ADT and Runtime Analysis Assessment Senior Exit Survey Alumni Survey Employer Survey

SLO 8: Students will be able to solve programming problems.	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. MFT Assessment Program Problem Solving Assessment ADT and Runtime Analysis Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 9: Students will demonstrate oral and written communication skills necessary to read, write, and speak effectively about concepts in computing.	Satisfied. Evidence to support the satisfactory completion of this learning outcome by our students is contained in the following assessments. Oral Communication Assessment Written communication Assessment Senior Exit Survey Alumni Survey Employer Survey
SLO 10: Students will understand ethical and legal issues involving digital technology.	Not Satisfied Analysis of the Ethical Assessment indicated that we needed some curriculum changes and a better method of assessing this learning objective. We have created a new three credit course to replace our one credit Computer Ethics class in our curriculum. The fall 2013 cohort will be the first set of students with this graduation requirement.

Satisfactory completion of a SLO assessment does not imply that we are not making improvements to our curriculum in these areas. It just means that the results indicate that we are enabling our students to demonstrate an understanding of these concepts and put them appropriately into action. We are continually looking for ways to improve our curriculum and our students' abilities and education.

Appendix

Major Field Test -- Computer Science (4HMF)

Item Information Report

Administration Date Range: April 2012 -- April 2014

Bloomsburg University

Number of Test Takers = 48

Section	Item Number[a]	Percent Correct Institution	Percent Correct National[b]	Percent Omit	Percent Not Reached	Domain	Content Area
1	1	35.4	22.2	0	0	Programming	Programming fundamentals
1	2	27.1	26.2	2.1	0	Discrete structures	Functions/relations/sets
1	3	62.5	61.0	0	0	Software engineering	Req/spec/design/val/mgmt
1	4	79.2	56.7	0	0	Algorithms/complexity	Automata theory
1	5	60.4	52.0	0	0	Systems	Architecture
1	6	81.2	68.9	0	0	Programming	Programming fundamentals
1	7	89.6	79.7	0	0	Programming	Programming fundamentals
1	8	35.4	35.1	0	0	Discrete structures	Counting/number theory
1	9	43.8	43.0	2.1	0	Other	Graphics
1	10	47.9	30.9	2.1	0	Systems	Operating systems
1	11	68.8	49.2	0	0	Algorithms/complexity	Basic computability/complexity
1	12	62.5	60.4	0	0	Programming	Programming languages
1	13	33.3	43.6	0	0	Algorithms/complexity	Adv data str/algorithms
1	14	87.5	63.4	0	0	Programming	Programming languages
1	15	33.3	22.0	0	0	Discrete structures	Basic logic
1	16	89.6	77.4	0	0	Information management	Data modeling
1	17	47.9	45.1	0	0	Systems	Operating systems
1	18	93.8	58.5	0	0	Programming	Programming fundamentals
1	19	35.4	38.7	0	0	Algorithms/complexity	Adv data str/algorithms
1	20	37.5	41.9	0	0	Discrete structures	Graphs/trees
1	21	12.5	11.7	0	0	Systems	Architecture
1	22	91.7	72.3	0	0	Information management	Database systems
1	23	35.4	27.8	0	0	Systems	Operating systems

Section	Item Number[a]	Percent Correct Institution	Percent Correct National[b]	Percent Omit	Percent Not Reached	Domain	Content Area
1	24	70.8	64.1	0	0	Algorithms/complexity	Adv data str/algorithms
1	25	95.8	87.4	0	0	Discrete structures	Basic logic
1	26	33.3	30.0	0	0	Systems	Networking
1	27	50.0	51.0	0	0	Software engineering	Req/spec/design/val/mgmt
1	28	25.0	24.3	0	0	Systems	Architecture
1	29**	----	----	----	----	----	----
1	30	66.7	45.1	2.1	0	Programming	Programming languages
1	31	25.0	27.4	0	0	Algorithms/complexity	Automata theory
1	32**	----	----	----	----	----	----
1	33	60.9	61.4	0	4.2	Discrete structures	Proof techniques
2	1	33.3	55.6	0	0	Software engineering	----
2	2	52.1	43.2	0	0	Discrete structures	Discrete probability
2	3	14.6	15.0	0	0	Algorithms/complexity	Basic computability/complexity
2	4	72.9	60.7	0	0	Programming	Programming fundamentals
2	5	64.6	50.3	0	0	Systems	Architecture
2	6	29.2	28.5	0	0	Systems	Architecture
2	7	43.8	35.2	0	0	Other	Intelligent systems
2	8	64.6	48.9	0	0	Discrete structures	Functions/relations/sets
2	9	70.8	58.2	0	0	Systems	Architecture
2	10	87.5	72.7	0	0	Programming	Programming fundamentals
2	11	16.7	27.0	0	0	Discrete structures	Functions/relations/sets
2	12	10.4	11.7	0	0	Systems	Architecture
2	13	39.6	41.1	0	0	Information management	Database systems
2	14	33.3	35.6	0	0	Algorithms/complexity	Automata theory
2	15	50.0	35.2	0	0	Programming	Programming fundamentals
2	16	47.9	35.7	0	0	Systems	Operating systems
2	17	25.0	33.3	0	0	Algorithms/complexity	Adv data str/algorithms
2	18	37.5	34.7	0	0	Algorithms/complexity	Automata theory
2	19	81.2	62.5	0	0	Programming	Programming fundamentals

Section	Item Number[a]	Percent Correct Institution	Percent Correct National[b]	Percent Omit	Percent Not Reached	Domain	Content Area
2	20	47.9	43.7	0	0	Systems	Architecture
2	21	14.6	26.2	0	0	Algorithms/complexity	Adv data str/algorithms
2	22	72.9	53.7	0	0	Programming	Programming languages
2	23	60.4	55.3	0	0	Discrete structures	Counting/number theory
2	24	39.6	25.7	0	0	Programming	Programming languages
2	25	50.0	37.6	0	0	Discrete structures	Graphs/trees
2	26	14.6	29.9	0	0	Software engineering	Req/spec/design/val/mgmt
2	27	12.5	15.5	0	0	Algorithms/complexity	Automata theory
2	28	72.9	59.4	0	0	Information management	Database systems
2	29	14.6	20.8	0	0	Software engineering	---
2	30	43.8	32.3	0	0	Programming	Programming fundamentals
2	31	22.9	35.2	0	0	Algorithms/complexity	Adv data str/algorithms
2	32	55.3	49.3	0	2.1	Discrete structures	Basic logic
2	33	19.6	28.6	0	4.2	Systems	Networking

(a) The total Computer Science test consists of 66 items. Items not scored are denoted by a double asterisk "**".

(b) Based on Comparative Data population for this form. Data ranges in date from September 2011 thru June 2013.

There are 3 Assessment Indicators (A) .

A1 -- Programming and Software Engineering

A2 -- Discrete Structures and Algorithms

A3 -- Systems: Architecture/Operating Systems/Networking/Database

Major Field Test - Computer Science (4LMF)

Item Information Report

Administration Date Range: December 2015 - April 2016

Bloomsburg University

Number of Test Takers = 20

Section	Item Number[a]	Percent Correct Institution	Percent Correct National[b]	Domain	Content Area	Item Mapping
1	1	35	19.2	Programming	Programming fundamentals	A1
1	2	75	64.8	Programming	Programming fundamentals	A1
1	3	55	62.9	Discrete structures	Proof techniques	A2
1	4	70	59.6	Algorithms/complexity	Automata theory	A2
1	5	45	47.3	Systems	Architecture	A3
1	6	95	65.9	Information management	Data modeling	A3
1	7	85	74.6	Software engineering	Req/spec/design/val/mgmt	A1
1	8	45	35.1	Discrete structures	Counting/number theory	A2
1	9	30	41.7	Other	Graphics	--
1	10	50	46.4	Algorithms/complexity	Adv data str/algorithms	A2
1	11	90	64	Systems	Architecture	A3
1	12	70	62.1	Programming	Programming languages	A1
1	13	20	40.1	Algorithms/complexity	Adv data str/algorithms	A2
1	14	60	48.8	Algorithms/complexity	Automata theory	A2
1	15	85	73.3	Discrete structures	Basic logic	A2
1	16	90	79.2	Information management	Data modeling	A3
1	17	55	45	Systems	Operating systems	A3
1	18	95	51.4	Programming	Programming fundamentals	A1
1	19	35	35.1	Information management	Data modeling	A3
1	20	80	81.7	Programming	Programming fundamentals	A1
1	21	45	57.6	Other	Security	--
1	22	65	50.3	Programming	Programming fundamentals	A1
1	23	35	36.4	Systems	Operating systems	A3
1	24	70	66.3	Algorithms/complexity	Adv data str/algorithms	A2
1	25	60	56.3	Algorithms/complexity	Automata theory	A2
1	26	35	42.7	Discrete structures	Counting/number theory	A2
1	27	30	15.9	Algorithms/complexity	Adv data str/algorithms	A2
1	28	10	19.4	Systems	Architecture	A3
1	29	0	18.6	Systems	Architecture	A3
1	30	50	48.2	Programming	Programming languages	A1
1	31**	--	--	--	--	--
1	32	10	28.9	Algorithms/complexity	Adv data str/algorithms	A2

1	33	68.4	61.5	Discrete structures	Proof techniques	A2
2	1	35	45.1	Software engineering	--	A1
2	2	45	46.5	Discrete structures	Discrete probability	A2
2	3	5	14.4	Algorithms/complexity	Basic computability/complexity	A2
2	4	90	89.1	Programming	Programming fundamentals	A1
2	5	25	23.6	Programming	Programming fundamentals	A1
2	6	30	27.6	Systems	Architecture	A3
2	7	75	62.6	Programming	Programming languages	A1
2	8	65	50.4	Discrete structures	Functions/relations/sets	A2
2	9	85	65.7	Programming	Programming fundamentals	A1
2	10	45	48.1	Algorithms/complexity	Adv data str/algorithms	A2
2	11	65	56.7	Algorithms/complexity	Automata theory	A2
2	12	45	33.9	Information management	Data modeling	A3
2	13	45	38.8	Discrete structures	Discrete probability	A2
2	14	50	41.9	Programming	Programming fundamentals	A1
2	15	55	58.3	Programming	Programming fundamentals	A1
2	16	50	33.6	Systems	Operating systems	A3
2	17	60	62.9	Systems	Architecture	A3
2	18	90	68.5	Programming	Programming fundamentals	A1
2	19	35	27.9	Software engineering	Req/spec/design/val/mgmt	A1
2	20	45	42.2	Systems	Architecture	A3
2	21**	--	--	--	--	--
2	22	65	54.2	Programming	Programming languages	A1
2	23	50	45.2	Programming	Programming fundamentals	A1
2	24	25	30.1	Discrete structures	Graphs/trees	A2
2	25	25	25.7	Algorithms/complexity	Basic computability/complexity	A2
2	26	15	23.4	Discrete structures	Functions/relations/sets	A2
2	27	89.5	69.2	Programming	Programming fundamentals	A1
2	28	47.4	52.2	Information management	Database systems	A3
2	29	68.4	63.5	Other	Intelligent systems	--
2	30	36.8	30.9	Programming	Programming fundamentals	A1
2	31	21.1	34.1	Algorithms/complexity	Adv data str/algorithms	A2
2	32	57.9	47.6	Discrete structures	Basic logic	A2
2	33	21.1	24.4	Systems	Architecture	A3

(a) The total Computer Science test consists of 66 items. Items not scored are denoted by a double asterisk "**".

(b) Based on Comparative Data population for this form. Data ranges in date from September 2015 thru June 2016.

There are 3 Assessment Indicators (A).

A1 -- Programming and Software Engineering

A2 -- Discrete Structures and Algorithms

A3 -- Systems: Architecture/Operating Systems/Networking/Database

Appendix Assessment Rubrics

Student name: _____ **Evaluator Name: _____**

C++ Assessment Rubric

	UNSATISFACTORY 1	MARGINAL 2	GOOD 3	EXCELLENT 4	SCORE
Pointers, operations on linked data structures, memory management	There is little or no demonstrated understanding of how to perform dynamic memory allocation or manipulate pointers.	There are missing or grossly incorrect functions and/or obvious errors that could cause memory leaks.	There are subtle errors that could lead to memory leaks but all functions are implemented and functional.	There are 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+)	There is little or no demonstrated understanding of how to overload operators and/or how to invoke them.	There are significant gaps in knowledge of how to overload operators and/or how to invoke them. Operator+ does not meet the complexity requirement.	The operator overloading is generally correct, but the complexity requirement for operator+ is not met.	The required operators are correctly overloaded, and the complexity requirement for operator+ is met.	
Templates	No attempt to implement a template class.	Major errors resulting in a non-functional template class, e.g., a member function is not a template function.	No major errors; the template class can be instantiated and is functional.	No functional errors, and uses recommended 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.	Highly non-cohesive interface. 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	There are significant deviations from coding standards throughout.	There are significant deviations from coding standards. The code is	The code is generally easy to read, but in some cases there may be	The code is professionally written: neatly organized, easy to read and	

	Many parts of the code are undocumented, overly complex, and/or cannot be understood without judgment or guesswork.	disorganized or poorly documented, and difficult to understand in places.	insufficient documentation, unusual or inconsistent indentation, cluttered or overly complicated code, or other minor deviations from coding standards.	understand, with correct indentation, reasonable choices for identifiers, and internal documentation to explain non-obvious details of the logic or its implementation.	

Student Name: _____ **Evaluator Name: _____**

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 introduce generic types	Generic types are introduced, but there are many problems with their specifications and implementations.	Generic types are introduced and they are used correctly in most instances.	Generic types are introduced and the types are used correctly in all instances.	
Java Coding Style (Programs are available to check for coding style)	No style	Many style faults	Most style conventions are followed. Most identifier names are appropriate. Most constants declared correctly.	All coding follows standard style conventions. All identifier names are appropriate. All constants are declared correctly.	
JavaDoc Documentation	Minimal java documentation. Most methods are not completely commented.	Many methods are not correctly documented.	Most methods are commented correctly and completely.	Each method and class has appropriate descriptions. All meta tags are correctly completed.	
Code	Code does not execute.	Code executes, but many implemented methods do not perform correctly.	Most implemented methods perform correctly.	The entire program is correct. All methods are implemented correctly.	
Problem Solution	Many program requirements are not completed.	Most requirements are completed, but few are correct.	Solution is well done; most requirements are completed correctly.	All program requirements are completed. Program is easy to use.	

Student Name: _____ **Evaluator Name: _____**

ADT and Runtime Analysis Rubric

	UNSATISFACTORY 1	MARGINAL 2	SATISFACTORY 3	EXCELLENT 4	SCORE
Analysis of Iterative Algorithms	Less than 35% correct.	36 – 60% correct.	61-– 85% correct.	86 – 100 % correct.	
Analysis of Recursive Algorithms	Less than 35% correct.	36 – 60% correct.	61-– 85% correct.	86 – 100 % correct.	
Application of Critical Thinking to Choosing Appropriate ADTs, Data Structures, and Algorithms	Less than 35% correct.	36 – 60% correct.	61-– 85% correct.	86 – 100 % correct.	

Student Name: _____ **Evaluator Name:** _____

Writing 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 piece has at most two 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 or are not present. Relationships among ideas are not evident.	Ideas are present but often unrelated. Main points are not evident. Little flow through the composition exists.	Main points are evident and usually related in a logical fashion. Introduction and conclusion are present.	Subject is introduced. Main points are developed. Transitions are made. Conclusions follow from main points.	

Notes:

(1) Content must be graded separately.