

In this issue:

- 4. Use of Artificial Intelligence to Grade Student Discussion Boards: An Exploratory Study**
Stephen M. Rutner, Georgia College and State University
Rebecca A. Scott, University of North Carolina Wilmington
- 19. The Role of Task Value and Online Learning Strategies in an Introductory Computer Programming Course**
Pratibha Menon, California University of Pennsylvania
- 32. Comparison of Information Technology Professionals' Perception of Job Satisfaction and Inclusion by Gender: Insights for Recruitment and Retention of Female Students**
Leah Schultz, Tarleton State University
Mark Adams, Tarleton State University
- 45. Redesigning Assessment in the Computer Science Program at a Regional University**
Ernst Bekkering, Northeastern State University
Patrick Harrington, Northeastern State University
- 89. Against "Flexibility": Tightening the Cage of Academic Rigor with Instructors' Responsibility and Rationality Student Group Satisfaction Perceptions using**
Yaojie Li, Columbus State University
Jason Xiong, Appalachian State University
Jennifer Pitts, Columbus State University
Scott Hunsinger, Appalachian State University
- 97. Upskilling and Reskilling for the Future of Work: A Typology of Digital Skills Initiatives**
Guido Lang, Quinnipiac University
Tamilla Triantoro, Quinnipiac University

The **Information Systems Education Journal** (ISEDJ) is a double-blind peer-reviewed academic journal published by **ISCAP** (Information Systems and Computing Academic Professionals). Publishing frequency is six times per year. The first year of publication was 2003.

ISEDJ is published online (<https://isedj.org>). Our sister publication, the Proceedings of EDSIGCON (<https://proc.iscap.info>) features all papers, panels, workshops, and presentations from the conference.

The journal acceptance review process involves a minimum of three double-blind peer reviews, where both the reviewer is not aware of the identities of the authors and the authors are not aware of the identities of the reviewers. The initial reviews happen before the EDSIGCON conference. At that point papers are divided into award papers (top 15%), other journal papers (top 25%), unsettled papers, and non-journal papers. The unsettled papers are subjected to a second round of blind peer review to establish whether they will be accepted to the journal or not. Those papers that are deemed of sufficient quality are accepted for publication in the ISEDJ journal. Currently the target acceptance rate for the journal is under 40%.

Information Systems Education Journal is pleased to be listed in the Cabell's Directory of Publishing Opportunities in Educational Technology and Library Science, in both the electronic and printed editions. Questions should be addressed to the editor at editor@isedj.org or the publisher at publisher@isedj.org. Special thanks to members of ISCAP/EDSIG who perform the editorial and review processes for ISEDJ.

2022 ISCAP Board of Directors

Eric Breimer Siena College President	Jeff Cummings Univ of NC Wilmington Vice President	Jeffry Babb West Texas A&M Past President/ Curriculum Chair
Jennifer Breese Penn State University Director	Amy Connolly James Madison University Director	Niki Kunene Eastern CT St Univ Director/Treasurer
RJ Podeschi Millikin University Director	Michael Smith Georgia Institute of Technology Director/Secretary	Tom Janicki Univ of NC Wilmington Director / Meeting Facilitator
Anthony Serapiglia St. Vincent College Director/2022 Conf Chair	Xihui "Paul" Zhang University of North Alabama Director/JISE Editor	

Copyright © 2022 by Information Systems and Computing Academic Professionals (ISCAP). Permission to make digital or hard copies of all or part of this journal for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial use. All copies must bear this notice and full citation. Permission from the Editor is required to post to servers, redistribute to lists, or utilize in a for-profit or commercial use. Permission requests should be sent to Paul Witman, Editor, editor@isedj.org.

INFORMATION SYSTEMS EDUCATION JOURNAL

Editors

Paul Witman
Editor
California Lutheran
University

Thomas Janicki
Publisher
U of North Carolina
Wilmington

Donald Colton
Emeritus Editor Brigham
Young University
Hawaii

Dana Schwieger
Associate Editor
Southeast Missouri
State University

Ira Goldman
Teaching Cases
Co-Editor
Siena College

Michelle Louch
Teaching Cases
Co-Editor
Carlow College

Brandon Brown
Cyber Education
Co-Editor
Coastline College

Anthony Serapiglia
Cyber Education
Co-Editor
St. Vincent College

Redesigning Assessment in the Computer Science Program at a Regional University

Ernst Bekkering
bekkerin@nsuok.edu

Patrick Harrington
harringp@nsuok.edu

Mathematics and Computer Science
Northeastern State University
Tahlequah, OK 74464

Abstract

This paper describes the development of the assessment program for the Computer Science program at a regional university in the Southwest. Assessment has become a major component in higher education. It is both a requirement for funding and accreditation, a signal of the quality of education provided, and a tool for improvement. Starting with an assessment program consisting of a single measurement at the end of the program, we have developed an assessment program that monitors student learning continuously throughout the curriculum. The new program allows for faster and more accurate identification of potential for improvement. The process of assessment program conversion and results from the first year are discussed in this paper.

Keywords: learning outcome assessment, learning outcomes, measurement, outcome-based education, program outcomes, accreditation.

1. INTRODUCTION

At the national level, outcomes assessment has gained attention through the "No Child Left Behind" (Boehner, 2002) and "Race to the Top" (U.S. Department of Education, 2016) programs. This movement has expanded to higher education (American Association of University Professors, 2015), where regional accrediting agencies, school accreditation agencies, and program accreditation agencies alike have instituted assessment as one of the core accreditation requirements. So far, only university-wide accreditation is a requirement to receive federal funding, whereas school accreditation and program accreditation are still voluntary. University accreditation may be the minimum standard, but assessment is a required component of all levels of accreditation.

Despite diverting attention from teaching and research, assessment is an opportunity for faculty to develop new models of assessment and analyze and use its results in a way that results for implementing meaningful change. This compares to formal definitions of assessment, which clearly include "closing the loop." The following definitions of assessment clearly include action at the end of the cycle. Osters (2003) defines assessment as "a systematic and ongoing process of collecting, interpreting, and acting on information relating to the goals and outcomes developed to support the mission and purpose of an institution." Likewise, Suskie (2018) describes assessment as "a cyclical, outcome-oriented process where faculty develop clearly written, measurable, and achievable program outcomes which are then systematically measured and evaluated with the results driving student learning and program improvement." It is the

latter activities that make assessment meaningful, but also the most difficult (Blaich & Wise, 2011; Fulcher et al., 2014).

In the United States, students in four-year programs typically spend the first two years in General Education (Gen Ed) courses, and complete major/minor specific courses in the junior and senior years. Assessment is important at both levels, and they can intersect. Outcome assessments for general education are embedded in Gen Ed courses, and occasionally in major/minor courses. For instance, communication can be measured in COMM 1113 Fundamentals of Oral Communication, but also in the capstone presentation of CS 4233 Professional Development in Computer Science (CS capstone). University-wide committees are responsible for assessment at the General Education level, whereas department-level committees conduct the program-specific assessment.

This paper is organized as follows. In the next section, we discuss relevant literature. In section 3, we discuss the development of our assessment program. Results of the first year's data collection are discussed in section 4. We end with conclusions and recommendations.

2. LITERATURE REVIEW

Assessment takes place at multiple levels. In this paper we describe assessment in one specific academic program, Computer Science. There are broader implications for our work that extend beyond assessment in one department.

Our university's assessment plan, last updated in 2017, is grounded in the university mission and vision. It aligns to the institutional priorities and strategic goals as articulated in the strategic plan. At this level, Degrees of Excellence (DOE) serve as foundation for assessment at the institutional level. They include intellectual skills, integrative knowledge, specialized knowledge in the major, integration of knowledge in the major with degree expectations, and citizenship. The degrees build upon the Lumina Foundation's Degree Qualification Profile (Lumina Foundation, 2021) and the Essential Learning Outcomes of the Association of American Colleges and Universities (Association of American Colleges & Universities, 2020).

Types of accreditation

Accreditation takes place in six regional accreditation agencies at the university, school, and program levels. These agencies include

accrediting bodies such as the regional NEASC and HLC, school-level ACBSP, and ABET for programs and majors.

The United States has six regional accreditation agencies (Figure 1). The Higher Learning Commission is a regional accreditation agency is one them (New England Commission of Higher Education, 2021). It accredits granting institutions of higher education based in the 19-state North Central region. Members are evaluated against five criteria (Higher Learning Commission, 2021):

1. Mission
2. Integrity: ethical and responsible conduct
3. Teaching and learning: quality, resources, and support
4. Teaching and learning: evaluation and improvement
5. Resources, planning, and institutional effectiveness.

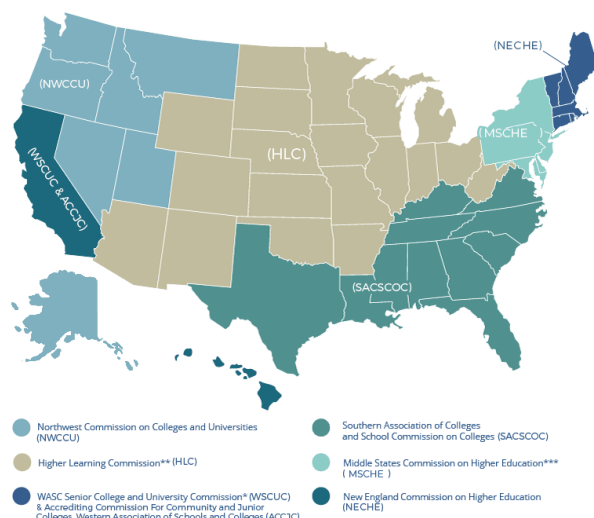


Figure 1: Regional Accreditation Agencies.
Source:chea.org

Program level assessment falls under the fourth criterion.

Program assessment is also required by the state's Regents for Higher Education. Regents expect assessment in four categories:

- Entry level to determine academic preparation and course placement
- General education assessment
- Academic program learning outcomes assessment
- Student engagement and satisfaction.

Notice that again, assessment is required at the program level but not at the course level within the programs.

Benefits of accreditation

Since accreditation demands sometimes significant effort and expense, it must be done in a way to maximize benefit. Most literature mentions indirect benefits, like stakeholder confidence, which cannot be directly measured. However, accreditation makes a program not only more attractive to potential students but can help with certifying excellence at a measurable level. At a more ordinary level, a direct benefit is that accreditation enables obtaining federal funds like student aid (Eaton, 2015). Financial Aid in all forms continues to climb from year to year (Educationdata.org, 2021), a trend higher education experts hope will remain (Washington Post, 2021). The Federal Government in return, pressured accreditation agencies to adopt an outcomes-based approach (Banta, 2001).

These benefits do not come without a price. Institutions face financial costs to the accrediting agency, burdens on staff and faculty, and potential threats to academic freedom (uPlanner, 2016). Faculty are frequently concerned about the impact of assessment on their workload (Hogan et al., 2002). At the program level, faculty may not have much support for accreditation either. Hilton et al. (2004) found the understanding of potential benefits of accreditation to be quite low among program administrators. Finally, accreditation can cause a loss of flexibility for program improvement. For example, ACBSP accreditation in another college, our College of Business, requires a common set of business courses for all BS degrees for a total of 42 hours, which limits the number of courses Information Systems (IS) students can take within their major. Overall though, the benefits of accreditation outweigh the limitations (Babb & Abdullat, 2014) and accreditation requirements are applicable to all programs dedicated to continuously delivering a quality curriculum, whether they are accredited or not (Challa et al., 2004).

Course vs. program outcomes assessment

In higher education, we can distinguish three levels of learning outcomes: course outcomes, program outcomes, and institutional outcomes. Occasionally, assessment at the school level is mentioned (Al-Mubaid et al., 2011). Course level outcomes are formal statements that describe what students are expected to learn in a course. They refer to specific knowledge, practical skills, areas of professional development, attitudes, higher-order thinking skills, etc. that faculty members expect students to develop, learn, or master during a course (Suskie, 2018). They can also be referred to as “learning outcomes”,

“student learning outcomes”, or “learning outcomes statements.”

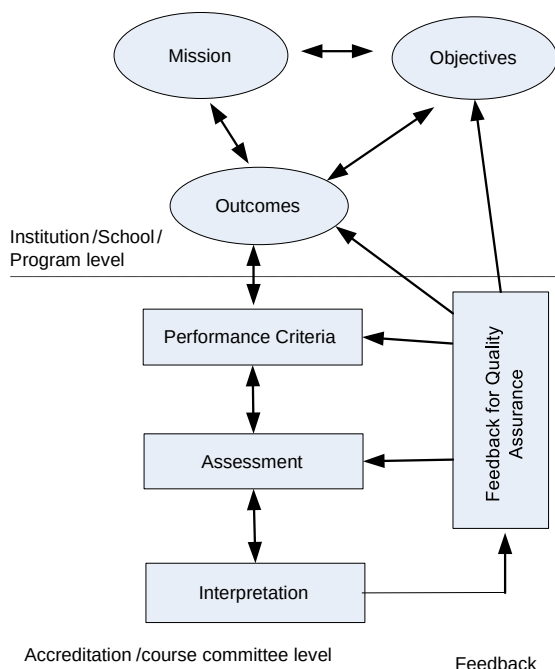


Figure 2: Assessment Model

The relationship between learning outcomes and the assessment process is discussed by Al-Mubaid et al. (2011). It consists of three components: guidance components (mission, objectives, and outcomes), evaluation components (performance criteria, performance of assessment processes, and interpretation of results), and feedback (Figure 2).

Whereas course level outcomes are narrow in scope (students achieve the outcomes as they complete a single course), program level outcomes are broad in scope (students achieve them as they complete the program at time of graduation). Lending et al. (2019) mention commitment to continuous assessment as one of the five essential ingredients of high-quality majors. For both course and program level assessment, it is important to use verbs that describe accurately what students will be able to do. Traditionally, higher education uses action verbs to indicate the level of performance (Anderson et al., 2001; Bloom, 1956).

The Institutional Effectiveness Plan (Figure 3) shows the relationship between the different types of outcomes.

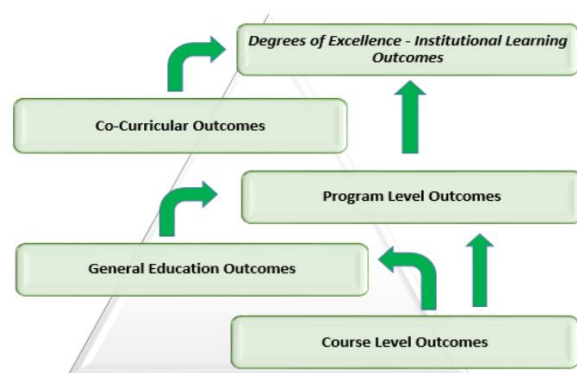


Figure 3: Outcome alignments

Outcomes measurement

Useful outcomes must be specific and measurable (Spady and Marshall 1994).

Program learning outcomes specify what students will know or be able to do at completion of the program. They need to meet the S.P.A.M criteria:

- Specific – start with a verb and target a single key competency
- Purposeful – relevant to students and the program
- Attainable – can be completed within a reasonable time
- Measurable – either direct or indirect measurement.

Course grades are not good measure of meeting goals because letter grades or scores do not closely correlate with strengths and weaknesses of individual learning outcomes. Rubrics, on the other hand, can combine the individual elements of multiple outcomes to a single score.

Types of measurement

Depending on the learning outcome, measurements can take multiple forms. We will briefly cover measures relevant to our program.

Measurements can use student work directly or indirectly. Direct measures use actual student work to produce evidence of student performance in the learning outcomes. Examples include performance assessments, capstone projects, senior theses, exhibits or performances, and standardized exams. Indirect measures use secondary information that do not rely on actual student work. Examples include satisfaction surveys, exit interviews, and focus groups. DFIW rates (proportion of D, F, Incomplete, Withdrawn grades) can serve to identify potential barriers to student success (Suskie, 2018).

Direct measures can be embedded in single courses or used across courses. The advantage of

embedded assessment is the high level of student motivation and the reduced burden on faculty to collect and analyze data. Course grades and test scores are not appropriate measurements, because grades give specific students feedback about performance, and assessment measures performance for all students as a group. Specific test questions or parts of larger work products can be used if they directly affect a Program Learning Outcome (PLO).

Capstone projects are good examples of cross-course measures. Projects combine the skills learned in multiple courses. A working program may be evidence of programming skills, and the documentation may be evidence of communication skills.

Indirect measures are harder to use and potentially more problematic. Employer surveys are only useful if respondents have first-hand knowledge of student work. Focus groups represent a limited number of opinions. Exit interviews with graduating students may not produce honest responses while still studying, and collection of data is harder after graduation. Perhaps the greatest use is not an assessment of how the program meets current students' needs, but to provide guidance about the desired level of performance after graduation.

Summative vs. formative assessment

Finally, the timing of assessment is relevant. Assessment can take place at the end of a single course or major program, but also during. Formative assessment takes place during the learning activity and benefits current students because there is still time for corrections. At the course level, an example might be a draft of a paper. At the program level, assignments in programming classes identify strengths and weaknesses in programming skills. Summative assessment takes place at the end when current students can no longer benefit but future students can. It evaluates courses and programs as a whole. For instance, capstone projects can be used to measure a wide variety of program objectives (Murray et al., 2008). In the examples above, the finished paper and capstone project can be used for summative assessment (Suskie, 2018, p. 157). For the health of a program, formative and summative assessments are equally important.

3. METHODOLOGY

This section describes the development of the current assessment plan for the Computer Science program.

Old assessment program

At the time of the last Higher Learning Commission accreditation visit in 2011, the CS program assessment was based on an exit exam from Educational Testing Services (2021) in paper and pencil form. Over time, we considered that the typical questions on the exam did not always reflect our program and attempted to devise a homegrown exit exam with typical questions from constituent courses in the program. Faculty members selected representative multiple-choice questions from core courses in the major, and after some discussion the final selection of questions was made.

Both tests suffered from two shortcomings, that of timing and relevance. Since they were administered one time only at the end of the last semester, had no consequences for non-participation and poor performance, and reflected material from courses taken in the past, the performance on these tests was low. Moreover, it was impossible to draw any conclusions that might lead to actionable results. Changes in the curriculum were the result of discussions among faculty, and not based on measurement of student learning.

New program

In 2017, the new university assessment plan was initiated. The plan defines steps in the learning outcomes assessment process as follows:

1. Articulate program mission and vision
2. Establish program learning outcomes
3. Develop outcome measures
4. Align program PLOs, outcome measures, and curriculum
5. Engage learners
6. Gather and analyze data
7. Share and report data
8. Close the loop by making evidence-based decisions

With this framework in mind, the process of preparing the assessment function for the next Higher Learning Commission accreditation visit started.

Computer Science program mission

First, we discussed our mission and vision statements in department meetings and made some slight revisions. The new mission statement became "The Department of Mathematics and Computer Science at (our university) will offer mathematics and computer science courses of the highest quality to both majors and non-majors through the faculty's dedication to excellence in teaching, commitment to scholarship, and determination to bring new knowledge into the

classroom so that each student will be prepared for advancement in his or her own field of study."

Assessment plan template

The assessment plan template went through minor revisions at the administrative level. In the Spring of 2019, the current assessment plan template was introduced by the Executive Director of Planning and Assessment. At the program level, it includes the mission statement, PLOs, a curriculum map that connects the PLOs with program courses and experiences, an assessment map that connects the PLOs with assessment measures and their performance targets, and a sample plan to increase consistency between programs.

Program Learning Outcomes

Prior to the start of the Fall 2019 semester, faculty and staff attended a keynote presentation by an experienced assessment speaker, and spent the rest of the day collaborating and discussing program goals and assessment instruments. Program Learning Outcomes were reviewed and revised to make them more specific, specifically to make them more measurable. **Table 1** compares the old and new program outcomes. In general, we replaced the verb "demonstrate" with a Bloom's action verb appropriate for the performance level we wished to pursue. We also included algorithms as one of the major components of computer science, to be measured at two different levels considering their importance.

Outcome measures

In developing our new assessment program, we moved away from one-time measurement to measuring student learning throughout the program. Program level outcomes must be addressed throughout the curriculum, and students need to have at least two courses where they can work towards these goals (Suskie, 2018, p. 55). The new assessment template reflected this distributed measurement. Over the course of the curriculum, students progress to the PLO in different stages. When the PLO is introduced (I), students acquire basic disciplinary knowledge and skills. Course activities help them to remember and/or understand. When the PLO is reinforced (R), students integrate skills with increasing complexity. Course activities give them the opportunity to apply and/or analyze. Finally, when the PLO is mastered (M), knowledge and skills are applied to address complex disciplinary questions and problems. Course activities focus on the advanced ability to evaluate and create.

Old Program Outcomes	New Program Outcomes
Demonstrate competency in multiple modern programming languages	Program in two or more computer languages
Demonstrate knowledge and understanding of the theory in core areas of Computer Science	Explain core elements of computer science theory
Demonstrate critical thinking and problem-solving skills through the application of software development process	Create software using software engineering principles
Demonstrate the ability to design and implement a computer system project	Solve problems by applying the software development process
Demonstrate the ability to use current techniques, skills, and tools necessary in the computing practice	Apply current techniques, skills, and tools necessary in the computing practice
Demonstrate the ability to learn new programming languages and acquire new skills independently	
Demonstrate effective written and oral communication skills	Communicate technical ideas both orally and in writing
Demonstrate the ability to collaborate effectively as a member of a team	Collaborate as a member of a team
	Design computer algorithms to solve problems
	Apply computer algorithms to solve problems

Table 1: Old and new program outcomes

Computer Science curriculum

Traditionally, Computer Science programs are rooted in either Electrical Engineering or Mathematics. Our university does not have an EE program, and the mathematics component is deliberately kept small so students can focus on programming skills. This is in line with proficiency in programming and software development, which is especially important in most CS

programs (Association of Computing Machinery, 2020). Some of the courses in the program are pulled from other majors where they are more appropriately taught, such as Technical Writing, which is taught in the English department. Table 2 presents a list of required and elective courses.

CS General Education Requirements	
ENGL 1113	English Composition I
ENGL 1213	English Composition II
MATH 1513	College Algebra
MATH 3023	Discrete Mathematics
CS Major Courses (31 hours)	
CS 2014	Computer Science I
CS 2163	Computer Science II
CS 3033	Object Oriented Programming
CS 3173	Basic Computer Architecture
CS 3343	Computer Operating Systems
CS3403	Data Structures
CS 4343	Database Management Systems
ENGL3083	Professional & Technical Writing
CS 4203	Software Engineering
CS 4233	Professional Development in CS
Select CS Electives (11 hours)	
CS 3023	Object Based Visual Programming
CS 3100	Select Topics
CS 3203	Application Development in C++
CS 3643	Programming for Cyber Security
CS 3663	Principles of TCP/IP
CS 4100	Advanced Select Topics
CS 4113	Computer Science Internship
CS 4143	Web Applications
CS 4223	Game Programming
CS 4253	Windows Programming
CS 4553	Parallel Programming

Table 2: CS Courses

Alignment with curriculum

Aligning PLOs, outcomes measures and curriculum makes the connection between what students will learn, how they will learn it, and how students demonstrate that they learned it. For instance, collaboration in a team requires team activities, and students can either rate each other or themselves (self-evaluation). Measuring team collaboration with a multiple-choice or essay test would not be appropriate. Writing software programs is best evaluated using actual programming assignments, and we developed a rubric that can be used to score essential elements of software programs (Appendix E>Error! Reference source not found.).

In the process of alignment, faculty decide which courses will be used for measuring specific

outcomes. In Information Systems, this process has been described by Veltri et al., (2011). Distribution of outcomes measurement over courses in the program is paramount. In our first draft of the curriculum map, the capstone course was assigned eight of nine PLOs, and three other courses only one. Not only did this provide a disproportionate burden on the faculty teaching the capstone course, but the measurement was done at a point in the program where few changes could be made if the students did not perform as expected. In the revised version, assessment activities were much more evenly distributed. Faculty needed to include no more than three assessment measurements in a course, and the total number of measurements was cut in half. This does not mean that students could not pursue learning outcomes in other courses; those specific outcomes were just not measured for assessment purposes. Appendix C shows the old and new curriculum mapping.

Next came development of the assessment measures. Since we had significant investment in development of the departmental exit exam, we started with its multiple-choice questions. We assigned them to the course from which they came, usually at the introduction level for the appropriate outcome. This resulted in nine outcome/course combination levels covered. We developed four rubrics: programming assignments, self-evaluation of group participation, oral communication, and capstone project evaluation. For the remainder of the measurements, we developed additional multiple-choice questions. In the process, we discovered that all relevant material was covered in outcomes 1 and 3-9, and that outcome 2 (Apply current techniques, skills, and tools necessary in the computing practice) was superfluous. We decided that it could be deleted and removed it from the program outcomes. Appendix A lists all outcomes at different levels for assessment. Appendix D provides a listing of all measurement items in the Computer Science program.

Data collection and analysis

All CS faculty are involved in collecting data in their respective courses and reporting the results to the Assessment Coordinator. All materials, from assessment instruments to data files and reports, past and present, have been stored in a shared folder so all faculty members have access. In addition, the Assessment Coordinator sends copies of the instrument for specific courses, which may contain measurements for more than one learning outcome, to the faculty teaching the course. This is done at the start of the semester

and after the mid-semester break to ensure that faculty remember to include the measurements in their teaching and testing materials.

If an instrument does not fit the course material, the instructor and assessment coordinator can make changes that don't substantively change the items. For instance, code fragments in Data Structures can be in C# in one course, and in Python in another depending on the language used.

After the semester, faculty email results to the assessment coordinator. Results are either in aggregate form (summary, averages) or anonymized (names replaced by numbers) to comply with the Family Educational Rights and Privacy Act (FERPA) (U.S. Department of Education, 2020). FERPA requires all student work to be maintained in a secure system with access limited to those involved in assessment, or personally identifiable information should be removed.

According to the assessment handbook, students should also be aware that their work may be used in assessment purposes. One way to do this is by incorporating a statement in select or all course syllabi. In the preparation for this article, we noticed that this was not common practice and reported this to Academic Affairs. As of this writing, it is not sure if a statement may be included in the campus-wide common syllabus template or not. If this is not the case, CS faculty have the option of including it in their individual syllabi.

As discussed before, each learning objective is measured two or three times in the program. Since students take courses in a predictable sequence due to prerequisites and course rotations, we can label them as introduced, reinforced, and mastered. The learning objectives are labeled as introduced in the first course where measured, mastered in the last course where measured, and reinforced if measured in between. This also allows us to set different performance targets based on placement in the sequence. Performance targets were arbitrarily set at 60% for introduced, 70% for reinforced, and 80% for mastered, to allow for improved mastery throughout the program and realistic program goal evaluation. Alternative goals, such as the % of students achieving at least 70% on an item (e.g. Aasheim et al., 2007) do not allow for improvement along the program.

Reporting data

The assessment program is cyclical in nature and cycles with the academic year (Figure: 4). Data are collected and analyzed in Fall and Spring semesters, the assessment report due September 1st is prepared at the end of Spring or in Summer, and the assessment plan is revised as needed at the start of the Fall Semester. Before submission by September 1st, CS faculty review the results, discuss potential changes to assessment instruments and the CS program, and submit the report to Academic Affairs.

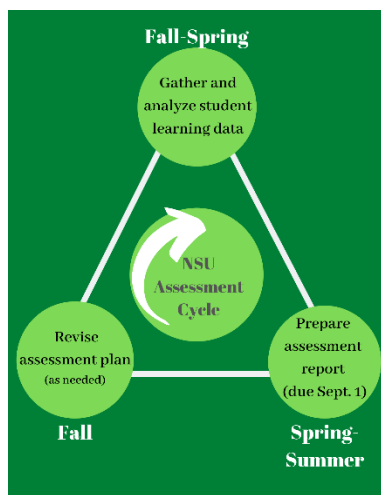


Figure: 4 Assessment cycle

The annual assessment report template, developed by the Office of Academic Affairs in collaboration with the Office of Institutional Effectiveness, is used across campus and includes the outcomes measures, the performance targets, the number of students assessed, the results, analysis of the results, and plans for the future (closing the loop).

4. FIRST YEAR'S RESULTS

Collecting and analyzing data for two semesters provided excellent results with regard to the current performance of our CS program. We will now describe the results for the eight different program outcomes. The specific measurement items are listed in Appendix D. Numbers of students and scores are listed separately for Fall and Spring.

Student communication

Our program goal "Communicate technical ideas both orally and in writing" was measured twice. In the ENGL3083 Technical writing course, students (n=5,13) scored well above the performance target of 60% with 92.5% and 82.5%. In the capstone oral presentation (n=14,

22) students scored initially below the performance target of 80% with 69.4%. After more thorough instruction in presenting skills during the capstone course, the scores improved to 85.5%

Computer programming

The second goal, "Program in two or more computer languages", was measured at three levels. At the introductory level, students (n=15, 8) surpassed the target of 60% with 92.4% and 92.9%. At the reinforced level, the goal was measured in three different courses in the fall (n=11, 20, 16) and once in the spring surpassing the 70% target with 89.3%, 87.6%, 94.0%, and 96.4%. At the mastered level, with a target of 80%, students (n=18, 30) scored 89.4% and 94.6%.

Designing algorithms

The third goal, "Design computer algorithms to solve problems", was again measured at three levels. At the introductory level, students in the fall (n=17) scored 70.6% with a goal of 60% and in two spring sections (n=17,15) 55.3% and 62.6%. At the reinforced level, students (n=15,8) fell short of the 70% mark with 33.0% and 40.6%. At the mastered level, data was only collected in the spring (n= 40, score = 92.0%, target 80%).

Applying algorithms

The fourth goal, "Apply computer algorithms to solve problems", was measured at three levels. At the introductory level, students (n = 15, 8) first fell short of the 60% target and then surpassed it with 72.5%. At the reinforced level, data was only collected in the spring (n = 40, score = 74.3%, target 70%). At the mastered level again, data collection in the spring (n= 30) fell short of the 80% target with a score of 60.7%.

Using software engineering principles

The fifth goal, "Create software using software engineering principles", was supposed to be measured at three levels. At the introductory level, students in the fall (n=17) fell short of the 60% target with 52.9% and in two spring sections (n=17,15) results were mixed with 50.6% and 66.7%. No data was collected at the reinforced level in either semester because the instructor in the course neglected to collect data. At the mastered level, the students in the fall (n=37) surpassed the 80% target with 87.8% and fell short in the spring (n=16) with 65.6%.

Collaboration

The sixth goal, "Collaborate as a member of a team" was measured at two levels. At the

introductory level, students (n=15,13) surpassed the 60% target with 92.0% and 93.0%. At the mastered level, using the same instrument and an 80% target, the results were mixed with 75.0% (n=37) and 82.2% (n=16).

Core elements of CS

The seventh goal, "Explain core elements of Computer Science", was measured at three levels. At the introductory level, students in the fall (n=45) surpassed the 60% target with 86.7%. At the reinforced level, two sections (n=23,19) had mixed results for a 70% target with scores of 87.0% and 44.0%. In the spring (n=19), results again fell short with 67.9%. At the mastered level, students (n=23,25) surpassed the 80% target with scores of 87.0% and 92.0%.

Problem solving

The final goal, "Solve problems by applying the software development process", was measured at three levels. At the introductory level, 17 students in the fall surpassed the 60% target with 64.7% and two sections in the spring (n=17,15) surpassed the target with 68.2% and 69.3%. At the reinforced level, students in both semesters (n=37,16) fell short of the 70% target with 65.4% and 65.6%. At the mastered level, one section in the fall (n=9) and three sections in the spring (n=12,5,13) surpassed the 80% target with 83.8%, 85.8%, 86.7%, and 85.3%.

5. CONCLUSIONS AND RECOMMENDATIONS

Redesigning our program assessment for university accreditation was an involved process, but it has already resulted in a better understanding of the program's strengths and weaknesses. Based on analysis of the data above, we conclude that our students have very good programming skills (goal 2), understand the core elements of CS (goal 7), could benefit from more work with algorithms (goals 3 and 4), and need more practice in professional presentations (goal 1). The process of data collection has also improved. In some instances, more data could still be collected. Only one of two instructors in ENGL 3083 submitted data, one CS faculty member may not have understood the process in the first semester and neglected to submit data in the second semester after having been released from the tenure track. Not all instructors require capstone presentations yet.

Assessment is an ongoing effort, and we plan to continue actively working within the assessment cycle to improve the quality of the CS program. One of the key determinants of successful

continuing assessment efforts may be generating institutional support for changes in program structure and allocation of resources as identified by the assessment process. Based on this assessment cycle, we can already identify some potential program changes. We could merge CS 3173 Basic Computer Architecture and CS 3343 Computer Operating Systems to a single course to make room for an advanced programming course. We are also considering removing ENGL 3083 Professional and Technical Writing to make room for an advanced computer algorithms course. The writing component could be replaced by requiring submission of professional portfolios on a site like GitHub. Career Services could be included for development of resumes and interview skills. Finally, the data suggest that we need to come to a more consistent approach between faculty teaching the same course, especially in the capstone course.

Now that we have set up a basic framework for assessment, we can use this as a framework to address applying for program accreditation. Based on previous publications, the transition from one type of accreditation (institutional) to another (program) may be relatively straightforward, even if there are some differences (Babb & Abdullat, 2014; Feinstein et al., 2014; Hilton et al., 2004). Our CS program underwent a program review last Fall. Using the assessment results, the recommendations of the program review, and the 2020 ACM Curriculum for Computer Science (Association of Computing Machinery, 2020), we hope to revise the curriculum in preparation for program accreditation. Results of these efforts may be reported at a later date.

6. REFERENCES

- Aasheim, C., Gowan, J. A., & Reichgelt, H. (2007). Establishing an assessment process for a computing program. *Information Systems Education Journal*, 5(1), 23-25.
- Al-Mubaid, H., Abeysekera, K., Kim, D., Perkins-Hall, S., & Yue, K.-B. (2011). A Model for Long Term Assessment of Computing and Information Systems Programs. *Information Systems Education Journal*, 9(3), 59.
- American Association of University Professors. (2015). *No Child Left Behind Goes to College*. AAUP. <https://www.aaup.org/article/no-child-left-behind-goes-college>
- Anderson, L. W., Kratwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Rath, J., & Wittrock, M. W. (2001). A

- Taxonomy for Learning, Teaching, and Assessing—A revision of Bloom's Taxonomy of Educational Objectives*. Longman.
- Association of American Colleges & Universities. (2020, April 30). *Essential Learning Outcomes* [Text]. Association of American Colleges & Universities. <https://www.aacu.org/essential-learning-outcomes>
- Association of Computing Machinery. (2020). *ACM and IEEE-CS issue Computing Curricula 2020*. <https://www.acm.org/media-center/2021/march/computing-curricula-2020>
- Babb, J. S., & Abdullat, A. (2014). Communicating the Value of Program-Level Accreditation for Information Systems in the College of Business. *Information Systems Education Journal*, 12(3), 39.
- Banta, T. W. (2001). Assessing competence in higher education. In *Assessing student competence in accredited disciplines: Pioneering approaches to assessment in higher education* (pp. 1–12). Stylus Sterling, VA.
- Blaich, C., & Wise, K. (2011). From gathering to using assessment results. *National Institute for Learning Outcomes Assessment*.
- Bloom, B. S. (1956). Taxonomy of educational objectives. Vol. 1: Cognitive domain. *New York: McKay*, 20, 24.
- Boehner, J. A. (2002, January 8). *H.R.1 - 107th Congress (2001-2002): No Child Left Behind Act of 2001 (2001/2002)*. <https://www.congress.gov/bill/107th-congress/house-bill/1>
- Challa, C. D., Kasper, G. M., & Redmond, R. (2004). The Accreditation Process for IS Programs in Business Schools. *Journal of Information Systems Education*, 16(2), 207.
- Eaton, J. S. (2015). An Overview of U.S. Accreditation. Revised November 2015. In *Council for Higher Education Accreditation*. Council for Higher Education Accreditation. <https://eric.ed.gov/?id=ED569225>
- Educational Testing Service. (2021). *Major Field Tests: Computer Science*. https://www.ets.org/mft/about/content/computer_science
- Educationdata.org. (2021). *Financial Aid Statistics*. EducationData. <https://educationdata.org/financial-aid-statistics>
- Feinstein, D., Longenecker, B., & Shresthat, D. (2014). A Study of Information Systems Programs Accredited by ABET In Relation to IS 2010. *Information Systems Education Journal*, 12(3), 76.
- Fulcher, K. H., Good, M. R., Coleman, C. M., & Smith, K. L. (2014). A Simple Model for Learning Improvement: Weigh Pig, Feed Pig, Weigh Pig. Occasional Paper# 23. *National Institute for Learning Outcomes Assessment*.
- Higher Learning Commission. (2021). *Criteria for Accreditation*. <https://www.hlcommission.org/Policies/criteria-and-core-components.html>
- Hilton, T. S., Johnson, D. A., & Kasper, G. M. (2004). ABET Accreditation of MIS Programs in AACSB Schools. *The Proceedings of Information Systems Education Conference (ISECON)*, 21, 1–16.
- Hogan, T. R., Harrison, P. R., & Schulze, K. G. (2002). Developing and maintaining an effective assessment program. *ACM SIGCSE Bulletin*, 34(4), 52–56.
- Lending, D., Mitri, M., & Dillon, T. W. (2019). Ingredients of a high-quality information systems program in a changing IS landscape. *Journal of Information Systems Education*, 30(4), 266.
- Lumina Foundation. (2021). *Degree Qualifications Profile*. DQP. <https://www.learningoutcomesassessment.org/dqp/>
- Murray, M. C., Perez, J., & Guimaraes, M. (2008). A model for using a capstone experience as one method of assessment of an information systems degree program. *Journal of Information Systems Education*, 19(2), 197.
- New England Commission of Higher Education. (2021). *Regional Accreditation Overview*. NECHE. <https://www.neche.org/for-the-public/faq/regionally-accredited-overview/>
- Northeastern State University. (2017). *Institutional Assessment Plan*. <https://offices.nsuok.edu/AcademicAssessment/Institutionalassessmentplan.aspx>
- Osters, S. (2003, February). *Writing measurable Learning Outcomes Texas A&M 3rd Annual 2003 Assessment Conference, February*. Retrieved May 25, 2021. <http://www.gavilan.edu/research/spd/Writing-Measurable-Learning-Outcomes.pdf>
- Suskie, L. (2018). *Assessing Student Learning: A Common Sense Guide* (3rd edition). Jossey-Bass.

- uPlanner. (2016, January 27). *Common challenges for accreditation in higher education institutions*.
<https://uplanner.com/en/blog/desafios-para-acreditar-programas-de-educacion-superior/>
- U.S. Department of Education. (2016, July 19). *Race to the Top Fund* [Program Home Page; Programs]. US Department of Education (ED).
<https://www2.ed.gov/programs/racetothetop/index.html>
- U.S. Department of Education. (2020, December 15). *Family Educational Rights and Privacy Act (FERPA)* [Guides]. US Department of Education (ED).
<https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html>
- Veltri, N. F., Webb, H. W., Matveev, A. G., & Zapatero, E. G. (2011). Curriculum mapping as a tool for continuous improvement of IS curriculum. *Journal of Information Systems Education*, 22(1), 31.
- Washington Post. (2021, April 25). Washington pumped \$35 billion into emergency grants for college students. Here's how it's going. *Washington Post*.
<https://www.washingtonpost.com/education/2021/04/24/college-emergency-aid-financial-assistance/>

APPENDIX A

Listing of Institutional, Program, Course Learning Outcomes

Degrees of Excellence Institutional Learning Outcomes

1. Intellectual skills emphasizing analytic inquiry, information literacy, engaging diverse perspectives, quantitative fluency, and communication fluency.
2. Integrative knowledge emphasizing the ability to produce, independently or collaboratively, an investigative, creative, or practical work that draws on specific theories, evidence, tools, and methods from diverse perspectives.
3. Specialized knowledge in the major emphasizing student competency in the program outcomes of the major field(s) of study.
4. Capstone Experience in the Baccalaureate Degree emphasizing the integration of the major with baccalaureate degree expectations reflecting the intersection of academic and post-baccalaureate settings.
5. Citizenship emphasizing leadership and engagement, experiential learning, cultural foundations, and personal and career development.

General Education Learning Outcomes

1. Communicate effectively through writing, listening, speaking, and reading;
2. Recognize and analyze works in the humanities (literature, art, music, philosophy, and religion) as expressions used to communicate perspectives on the human condition;
3. Identify and evaluate political, historical, and social forces that shape the past, present, and future;
4. Become globally-aware citizens through an understanding and appreciation of human and cultural diversity;
5. Understand physical and biological phenomena and their importance for the welfare of society;
6. Apply methods of scientific inquiry;
7. Use quantitative symbolic systems to solve problems and interpret data;
8. Understand and apply concepts and activities that promote good health and life skills;
9. Use critical thinking to analyze and solve problems.

Computer Science Program Learning Outcomes

1. Communicate technical ideas both orally and in writing.
2. Apply current techniques, skills, and tools necessary in the computing practice (deleted).
3. Program in two or more computer languages.
4. Design computer algorithms to solve problems.
5. Apply computer algorithms to solve problems.
6. Create software using software engineering principles.
7. Collaborate as a member of a team.
8. Explain core elements of computer science theory.
9. Solve problems by applying the software development process.

APPENDIX B

Higher Learning Commission Criterion for Evaluation and Improvement of Teaching and Learning

Link: <https://www.hlcommission.org/Policies/criteria-and-core-components.html>

Criterion 4. Teaching and Learning: Evaluation and Improvement

The institution demonstrates responsibility for the quality of its educational programs, learning environments, and support services, and it evaluates their effectiveness for student learning through processes designed to promote continuous improvement.

Core Components

4.A. The institution ensures the quality of its educational offerings.

1. The institution maintains a practice of regular program reviews and acts upon the findings.
2. The institution evaluates all the credit that it transcripts, including what it awards for experiential learning or other forms of prior learning, or relies on the evaluation of responsible third parties.
3. The institution has policies that ensure the quality of the credit it accepts in transfer.
4. The institution maintains and exercises authority over the prerequisites for courses, rigor of courses, expectations for student learning, access to learning resources, and faculty qualifications for all its programs, including dual credit programs. It ensures that its dual credit courses or programs for high school students are equivalent in learning outcomes and levels of achievement to its higher education curriculum.
5. The institution maintains specialized accreditation for its programs as appropriate to its educational purposes.
6. The institution evaluates the success of its graduates. The institution ensures that the credentials it represents as preparation for advanced study or employment accomplish these purposes. For all programs, the institution looks to indicators it deems appropriate to its mission.

4.B. The institution engages in ongoing assessment of student learning as part of its commitment to the educational outcomes of its students.

1. The institution has effective processes for assessment of student learning and for achievement of learning goals in academic and cocurricular offerings.
2. The institution uses the information gained from assessment to improve student learning.
3. The institution's processes and methodologies to assess student learning reflect good practice, including the substantial participation of faculty, instructional and other relevant staff members.

4.C. The institution pursues educational improvement through goals and strategies that improve retention, persistence and completion rates in its degree and certificate programs.

1. The institution has defined goals for student retention, persistence and completion that are ambitious, attainable, and appropriate to its mission, student populations and educational offerings.
2. The institution collects and analyzes information on student retention, persistence, and completion of its programs.
3. The institution uses information on student retention, persistence, and completion of programs to make improvements as warranted by the data.
4. The institution's processes and methodologies for collecting and analyzing information on student retention, persistence and completion of programs reflect good practice. (Institutions are not required to use IPEDS definitions in their determination of persistence or completion rates. Institutions are encouraged to choose measures that are suitable to their student populations, but institutions are accountable for the validity of their measures.)

APPENDIX C Initial Draft and Final Version of Course Mapping

Degree Program Courses or Experiences	Communicate technical ideas both orally and in writing.	Apply current techniques, skills, and tools necessary in the computing practice.	Program in two or more computer languages.	Design computer algorithms to solve problems.	Apply computer algorithms to solve problems.	Create software using software engineering principles.	Collaborate as a member of a team.	Explain core elements of computer science theory.	Solve problems by applying the software development process.	
CS 2014 Computer Science I		introduced	introduced		introduced					3
CS 2163 Computer Science II		reinforced	reinforced	introduced	reinforced					4
CS 3033 Obj Oriented Programming		reinforced	reinforced	reinforced	reinforced	introduced		introduced		6
CS 3173 Basic Computer Architecture								introduced		1
CS 3343 Computer Operating Systems								reinforced		1
CS 3403 Data Structures		reinforced	reinforced	mastered	mastered	introduced		reinforced		6
CS 4203 Software Engineering	reinforced	mastered				reinforced	introduced	reinforced	introduced	6
CS 4233 Professional Development in CS	mastered	mastered	mastered	mastered	mastered	mastered	reinforced		reinforced	8
CS 4343 Database Management Systems	reinforced	mastered	reinforced	reinforced	mastered					5
ENGL 3083 Professional & Technical Writing	introduced									1
MATH 3023 Discrete Mathematics			reinforced	reinforced						2
	4	7	7	6	6	4	2	5	2	
									items	43

Degree Program Courses or Experiences	Communicate technical ideas both orally and in writing.	Apply current techniques, skills, and tools necessary in the computing practice.	Program in two or more computer languages.	Design computer algorithms to solve problems.	Apply computer algorithms to solve problems.	Create software using software engineering principles.	Collaborate as a member of a team.	Explain core elements of computer science theory.	Solve problems by applying the software development process.	
CS 2014 Computer Science I				introduced		introduced			introduced	3
CS 2163 Computer Science II			introduced	reinforced	introduced					3
CS 3033 Obj Oriented Programming			mastered	mastered	reinforced					3
CS 3173 Basic Computer Architecture							introduced	reinforced		2
CS 3343 Computer Operating Systems								mastered		1
CS 3403 Data Structures					mastered					1
CS 4203 Software Engineering						mastered	mastered		reinforced	3
CS 4233 Professional Development in CS	mastered								mastered	2
CS 4343 Database Management Systems						reinforced				1
ENGL 3083 Professional & Technical Writing	introduced									1
MATH 3023 Discrete Mathematics								introduced		1
any other programming elective (optional)			reinforced							1
	2	0	3	3	3	3	2	3	3	
									items	22

APPENDIX D
Measurement Items

CS 2014: Computer Science I

Design computer algorithms to solve problems (introduced)

State the output for each of the following program segments. Assume that there are no syntax errors and that the appropriate header statements are there to make it compile correctly.

1.

```
for(int i=1; i< 3; i++)  
{  
    cout << i << " ";  
}
```

- a) 1, 2, 3
- b) 1 3
- c) 3
- d) 1 2

2.

```
int myarray[3];  
for(int i=0; i< 3; i++)  
    myarray[i] = i;  
for(int i=0; i< 3; i++)  
    cout << myarray[i] << endl;
```

- a) 0, 1, 2
- b) 0 1 2
- c) 0 1
2
- d) 0
1
2

3.

```
int i=1, total=0;  
while(i < 4)  
{  
    total=total+i;  
    i++;  
}  
cout << total << " " << i << endl;
```

	i	total
	1	0
	2	1
	3	3
	4	6

- a) 6 3
- b) 6 4
- c) 10
- d) 4 6

4.

```
void dostuff(char a[], char b[]);
```

```
void main(void) {  
    char s1[] = "PAPOA";  
    char s2[] = "WEHLO";  
  
    dostuff(s1, s2);  
    cout << s1;  
}
```

```
void dostuff(char a[], char b[]) {  
    a[0] = 'Y';
```

```
a[2] = b[2];  
a[4] = b[4];  
}
```

- a) PAHAO
 - b) YAHOA
 - c) PAPOA
 - d) YAHOO
5. `int x=5, y=-1, &rx=x, &ry=y, z;`
`z = rx + ry;`
`cout << x << ", " << ry << ", " << z;`
- a) NULL
 - b) 5 -1 4
 - c) 5 -1 NULL
 - d) 5, -1, 4

Create software using software engineering principles (introduced)

1. Which of the following is not a nested loop?

- a) `for(i=0;i<10;i++)`
`for(j=1;j<i+2;j++)`
`cout << i*j << endl;`
- b) `for(i=0;i<10;i++)`
`cout<<"i="<<i;`
`for(j=1;j<i+2;j++) cout<<"j="<<j;`
- c) `for(i=0;i<10;i++)`
`while(j%2!=0) { cout<<j<<"\t"; j++; }`
- d) `while(j<x)`
`{`
`while(x<z)`
`x+=j;`
`}`

2. How many times will the loop print "NSU"?

```
#include <iostream>  
int x = 5;  
while (x = 5)  
{  
    cout << "NSU\n" << endl;  
    ++x;  
}
```

- a) infinite loop
- b) 4
- c) 5
- d) 0

3. Convert the While loop to a Do While loop

```
while (number <1 || number >10)
```

- ```
{
a) Do
{
```

```
 cout << "enter a number between 1 and 10: ";
cin >> number;
}while (number <1 || number >10);
```

b) Do;  
{  
 cout << "enter a number between 1 and 10: ";  
cin >> number;  
}while (number <1 || number >10)

c) Do  
{  
 cout << "enter a number between 1 and 10: ";  
cin >> number;  
}while (number <1 && number >10);

d) Do  
{  
 cout << "enter a number between 1 and 10: ";  
cin >> number;  
}while (number >1 || number >10);

4. Convert the While loop to a For loop

```
while(counter <= 5)
{
 cout << (startNumber + counter) << " squared is "
 << pow((startNumber + counter),2) << endl;
 counter++;
}
```

a) for(int counter = 1; counter <=5; counter++)  
{  
 cout << (startNumber + counter) << " squared is "  
 << pow((startNumber + counter),2) << endl;  
}

b) for(int counter = 1, counter <=5, counter++)  
{  
 cout << (startNumber + counter) << " squared is "  
 << pow((startNumber + counter),2) << endl;  
}

c) {  
 cout << (startNumber + counter) << " squared is "  
 << pow((startNumber + counter),2) << endl;  
} for(int counter = 1; counter <=5; counter++)

d) {  
 cout << (startNumber + counter) << " squared is "  
 << pow((startNumber + counter),2) << endl;  
} for(int counter = 1, counter <=5, counter++)

5. Which code will create a random roll of the die?

a) #include <cstdlib>  
 #include <ctime>

```
.....
const int MIN_VALUE = 1;
const int MAX_VALUE = 6;
unsigned seed = time(0);
srand(seed);
cout << "Rolling the dice...\n";
int die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
```

- b) #include <cstdlib>  
#include <ctime>

```
.....
const int MIN_VALUE = 1;
const int MAX_VALUE = 6;
unsigned seed = time(0);
srand(seed);
cout << "Rolling the dice...\n";
int die = (rand() % (MAX_VALUE - MIN_VALUE) + MIN_VALUE;
cout << die << endl;
```

- c) #include <cstdlib>  
#include <ctime>

```
.....
const int MIN_VALUE = 1;
const int MAX_VALUE = 6;
cout << "Rolling the dice...\n";
int die = (rand() % (MAX_VALUE - MIN_VALUE+1));
cout << die << endl;
```

- d) #include <cstdlib>  
#include <ctime>

```
.....
const int MIN_VALUE = 1;
const int MAX_VALUE = 6;
unsigned seed = time(0);
random(seed);
cout << "Rolling the dice...\n";
int die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
```

### **Solve problems by applying the software development process (introduced)**

1. Put the steps of software development in the proper order:
  - a. Develop algorithms in pseudocode
  - b. Gathering requirements
  - c. Testing
  - d. Write programming code

- a) b - a - d - c
- b) a - b - c - d
- c) c - d - a - b
- d) d - a - c - b

2. How can we prevent the input problem in this program?

```
#include <iostream>
using namespace std;
int main()
```

```
{
 int year;
 char ch;
 cout << "Enter the year you were born: ";
 cin >> year;
 cout << "Enter your middle initial: ";
 ch = cin.get();
 cout << ch << " was born in " << year << "\n";
 return 0;
}
/* output:
Enter the year you were born: 1988
Enter your middle initial:
was born in 1988
Press any key to continue . . .
*/
```

- a) add `cin.ignore()` after the input line for the year
- b) change `ch = cin.get()` to `cin >> ch`
- c) change `cin >> year` to `year = cin.get()`
- d) change `int year` to `char[4] year`

3. Put the steps to create an executable program in the proper order:
- a. run the compiler to convert source code to machine instructions
  - b. run the preprocessor to convert source file directives to source code program statements
  - c. run the linker to connect hardware-specific code to machine instructions
  - d. create a source code file with a text editor

- a) d - b - a - c
- b) a - b - c - d
- c) d - a - c - b
- d) b - c - d - a

4. Which loop will create the following output: 20 17 14 11 8 5

- a) 

```
for (int x = 20; x >= 5; x -= 3)
 cout << x << " ";
```
- b) 

```
for (int x = 20; x > 5; x -= 3)
 cout << x << " ";
```
- c) 

```
for (int x = 20; x < 5; x -= 3)
 cout << x << " ";
```
- d) 

```
for (int x = 20; x <= 5; x -= 3)
 cout << x << " ";
```

5. How can we BEST improve the following program?

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main()
{
 const int MIN_VALUE = 1;
 const int MAX_VALUE = 6;

 unsigned seed = time(0);
 srand(seed);
```

```
int die;

cout << "Rolling the dice...\n";
die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
cout << "Rolling the dice...\n";
die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
cout << "Rolling the dice...\n";
die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
cout << "Rolling the dice...\n";
die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;
cout << "Rolling the dice...\n";
die = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
cout << die << endl;

return 0;
}
```

- a) create a function
- b) change the constants to regular integers
- c) use global variables
- d) add input checking
- e) make die a constant

### CS 2163: Computer Science II

#### Program in two or more computer languages (introduced)

##### Programmer(s)

##### Program Assessment

**3 = Good    2 = Average    1 = Below Average    0 = Not gradable**

| Criteria                                                                             | Comments | Points |   |   |   |
|--------------------------------------------------------------------------------------|----------|--------|---|---|---|
| The program fully implements the solution to the given problem                       |          | 3      | 2 | 1 | 0 |
| The solution produces the correct result                                             |          | 3      | 2 | 1 | 0 |
| The program runs without error, has reasonable error checking and exception handling |          | 3      | 2 | 1 | 0 |
| The user is prompted for input when required                                         |          | 3      | 2 | 1 | 0 |
| The user is presented with appropriate feedback as to the results of an operation    |          | 3      | 2 | 1 | 0 |
| All project files and source code required to build and execute are submitted        |          | 3      | 2 | 1 | 0 |
| Code is properly indented, and adequately commented                                  |          | 3      | 2 | 1 | 0 |

|                                                                                                      |  |   |   |   |   |
|------------------------------------------------------------------------------------------------------|--|---|---|---|---|
| Code is well written and free from unnecessary complexity or redundancy                              |  | 3 | 2 | 1 | 0 |
| Programmer defined symbol names are reasonable (variable names, class names, function names, etc...) |  | 3 | 2 | 1 | 0 |
| The user interface is accurate and is free from misspelled words and bad grammar                     |  | 3 | 2 | 1 | 0 |
| <b>Total Points out of 30:</b>                                                                       |  |   |   |   |   |
| <b>Comments:</b>                                                                                     |  |   |   |   |   |

### Design computer algorithms to solve problems (reinforced)

1. State output for the following:

```
int fun3(int a[], int b[], int cnt);
int main(void) {
 const int TOP = 3;
 int x[TOP] = {5, 3, 2};
 int y[TOP] = {1, 4, 3};
 int val;

 val = fun3(x, y, TOP);
 for(int i=0; i<TOP; i++)
 cout << x[i] << ", ";
 cout << endl << val;
}
int fun3(int a[], int b[], int cnt)
{
 int val=0;
 for(int i=0; i<cnt; i++)
 {
 if(b[i] > a[i])
 a[i] = b[i];
 val += a[i];
 }
 return val;
}
```

- A. 5, 4, 3, 12
- B. 5, 4, 3, 12
- C. 5, 4, 3, 12
- D. 5, 3, 2, 1, 4, 3

2. Which of the following code will correctly begin a linked list to hold a double?

```
A.
struct ListNode
{
```

```
 double value;
 struct ListNode *next;
};
ListNode head;
```

B.

```
struct ListNode
{
 double value;
 struct ListNode *next;
};
ListNode *head;
```

C.

```
struct ListNode
{
 double value;
 struct ListNode next;
};
ListNode *head;
```

D. None of these answers are correct

3. Which of the following code will correctly append an element into a linked list?

A. void NumberList::appendNode(double num)

```
{
 ListNode newNode;
 ListNode nodePtr;
 newNode = new ListNode;
 newNode->value = num;
 newNode->next = NULL;
 if (!head)
 head = *newNode;
 else {
 nodePtr = head;
 while (nodePtr->next)
 nodePtr = nodePtr->next;
 nodePtr->next = newNode;
 }
}
```

B. void NumberList::appendNode(double num)

```
{
 ListNode *newNode;
 ListNode *nodePtr;
 newNode = new ListNode;
 newNode->value = num;
 newNode->next = NULL;
 if (!head)
 head = newNode;
 else {
 nodePtr = head;
 while (**nodePtr)
 nodePtr = *next;
 **next = *newNode;
 }
}
```

```
}
```

```
C. void NumberList::appendNode(double num)
{
 ListNode *newNode;
 ListNode *nodePtr;
 newNode = new ListNode;
 newNode->value = num;
 newNode->next = NULL;
 if (!head)
 head = newNode;
 else {
 nodePtr = head;
 while (nodePtr->next)
 nodePtr = nodePtr->next;
 nodePtr->next = newNode;
 }
}
```

D. None of these answers is correct

4. Given the following output:

enter an integer 10

that value doubled is 20

Which of the following programs below produces the above output?

```
A. void getNumber(int *input)
{
 cout << "enter an integer " ;
 cin >> *input;
}
void doubleValue(int *val)
{
 *val *= 2;
}

void main()
{
 int number = 0;
 getNumber(&number);
 doublevalue(&number);
 cout << "that value doubled is " << number << endl;
}
```

```
B. void getNumber(int *input)
{
 cout << "enter an integer " ;
 cin >> input;
}
void doubleValue(int *val)
{
 val *= 2;
}

void main()
```

```
{
 int number = 0;
 getNumber(&number);
 doublevalue(&number);
 cout << "that value doubled is " << number << endl;
}

C. void getNumber(int input)
{
 cout << "enter an integer " ;
 cin >> input;
}
void doubleValue(int val)
{
 val *= 2;
}

void main()
{
 int number = 0;
 getNumber(number);
 cout << "that value doubled is " << number << endl;
}
```

D. None of these answers are correct.

### **Apply computer algorithms to solve problems (introduced)**

1. Which algorithm is shown below?

```
int func(const int array[], int size, int value)
{
 int index = 0;
 int position = -1;
 bool found = false;

 while (index < size && !found)
 {
 if (array[index] == value)
 {
 found = true;
 position = index;
 }
 index++;
 }
 return position;
}
```

a) linear search  
b) bubble sort  
c) binary search  
d) selection sort

2. which algorithm is shown below?

```
void func(int array[], int size)
```

```
{
 bool swap;
 int temp;

 do
 {
 swap = false;
 for (int count = 0; count < (size - 1); count++)
 {
 if (array[count] > array[count + 1])
 {
 temp = array[count];
 array[count] = array[count + 1];
 array[count + 1] = temp;
 swap = true;
 }
 }
 } while (swap);
}
```

a) bubble sort  
b) binary search  
c) selection sort  
d) linear search

3. Which algorithm is shown below?

```
void func(int array[], int size)
{
 int startScan, minIndex, minValue;

 for (startScan = 0; startScan < (size - 1); startScan++)
 {
 minIndex = startScan;
 minValue = array[startScan];
 for(int index = startScan + 1; index < size; index++)
 {
 if (array[index] < minValue)
 {
 minValue = array[index];
 minIndex = index;
 }
 }
 array[minIndex] = array[startScan];
 array[startScan] = minValue;
 }
}
```

a) selection sort  
b) binary search  
c) bubble sort  
d) linear search

4. Which algorithm is shown below?

```
int func(const int array[], int size, int value)
{
 int first = 0,
 last = size - 1,
 middle ,
 position = -1;
 bool found = false;
```

```

while (!found && first <= last)
{
 middle = (first + last) / 2;
 if (array[middle] == value)
 {
 found = true;
 position = middle;
 }
 else if (array[middle] > value)
 last = middle - 1;
 else
 first = middle + 1;
}
return position;
}
a) binary search
b) selection sort
c) bubble sort
d) linear search

```

5. Which algorithm is shown below?

```

int func(int arr[], int l, int r, int x)
{
 if (r >= l) {
 int mid = l + (r - l) / 2;
 if (arr[mid] == x)
 return mid;
 if (arr[mid] > x)
 return func(arr, l, mid - 1, x);
 return func(arr, mid + 1, r, x);
 }
 return -1;
}
a) binary search
b) selection sort
c) bubble sort
d) linear search

```

### CS 3033: Object Oriented Programming

#### Program in two or more computer languages (mastered)

Programmer(s)

#### Program Assessment

**3 = Good    2 = Average    1 = Below Average    0 = Not gradable**

| Criteria                                                                             | Comments | Points |   |   |   |
|--------------------------------------------------------------------------------------|----------|--------|---|---|---|
| The program fully implements the solution to the given problem                       |          | 3      | 2 | 1 | 0 |
| The solution produces the correct result                                             |          | 3      | 2 | 1 | 0 |
| The program runs without error, has reasonable error checking and exception handling |          | 3      | 2 | 1 | 0 |
| The user is prompted for input when required                                         |          | 3      | 2 | 1 | 0 |

|                                                                                                      |  |   |   |   |   |
|------------------------------------------------------------------------------------------------------|--|---|---|---|---|
| The user is presented with appropriate feedback as to the results of an operation                    |  | 3 | 2 | 1 | 0 |
| All project files and source code required to build and execute are submitted                        |  | 3 | 2 | 1 | 0 |
| Code is properly indented, and adequately commented                                                  |  | 3 | 2 | 1 | 0 |
| Code is well written and free from unnecessary complexity or redundancy                              |  | 3 | 2 | 1 | 0 |
| Programmer defined symbol names are reasonable (variable names, class names, function names, etc...) |  | 3 | 2 | 1 | 0 |
| The user interface is accurate and is free from misspelled words and bad grammar                     |  | 3 | 2 | 1 | 0 |
| <b>Total Points out of 30:</b>                                                                       |  |   |   |   |   |
| <b>Comments:</b>                                                                                     |  |   |   |   |   |

### Design computer algorithms to solve problems (mastered)

1. Consider the program below:

```
public class Test
{
 public static void main(String[] args)
 {
 int[] a;
 a = new int[10];

 for (int i = 0; i < a.length; i++)
 a[i] = i + 2;
 int result = 0;
 for (int i = 0; i < a.length; i++)
 result += a[i];
 System.out.printf("Result is: %d%n", result);
 }
}
```

The output of this program will be:

- a) Result is: 62.
  - b) Result is: 64.
  - c) Result is: 65.
  - d) Result is: 67.
2. Assume class Book has been declared. Which set of statements creates an array of Books?

- a.
 

```
Book[] books;
books = new Book[numberElements];
```
- b.
 

```
Book[] books;
books = new Book()[numberElements];
```
- c.
 

```
new Book() books[];
books = new Book[numberElements];
```
- d. All of the above.

3. Which of the following class members should usually be private?
  - a) Methods.
  - b) Constructors.
  - c) Variables (or fields).
  - d) All of the above.
4. Static class variables:
  - a) are final.
  - b) are public.
  - c) are private.
  - d) are shared by all objects of a class.
5. Which of the following statement displays Hello World?
  - a) `System.out.printf( "%2s", "Hello " "World" );`
  - b) `System.out.printf( "%s %s", "Hello", "World" );`
  - c) `System.out.printf( "%s%s", "Hello, World" );`
  - d) `System.out.printf( "s% s%", "Hello", "World" );`
6. Which of the following statements is true?
  - a) Local variables are automatically initialized.
  - b) Every instance variable has a default initial value—a value provided by Java when you do not specify the instance variable's initial value.
  - c) The default value for an instance variable of type String is void.
  - d) The argument types in the method call must be identical to the types of the corresponding parameters in the method's declaration.
7. When must a program explicitly use the this reference?
  - a) Accessing a private variable.
  - b) Accessing a public variable.
  - c) Accessing a local variable.
  - d) Accessing an instance variable that is shadowed by a local variable.
8. A constructor *cannot*:
  - a) be overloaded.
  - b) initialize variables to their defaults.
  - c) specify return types or return values.
  - d) have the same name as the class.
9. Which superclass members are inherited by all subclasses of that superclass?
  - a) private instance variables and methods.
  - b) protected instance variables and methods.
  - c) private constructors.
  - d) protected constructors.
10. Overriding a method differs from overloading a method because:
  - a) Overloaded methods have the same signature.
  - b) Overridden methods have the same signature.
  - c) Overridden methods must have different signatures
  - d) Neither of the above.

### **Apply computer algorithms to solve problems (reinforced)**

1. Which algorithm is shown below?  
`public static int func(int arr[], int x)`

```
{
 int n = arr.length;
 for(int i = 0; i < n; i++)
 {
 if(arr[i] == x)
 return i;
 }
 return -1;
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

2. Which algorithm is shown below?

```
public static int func(int arr[], int l, int r, int x)
{
 if (r >= l) {
 int mid = l + (r - l) / 2;
 if (arr[mid] == x)
 return mid;
 if (arr[mid] > x)
 return func(arr, l, mid - 1, x);
 return func(arr, mid + 1, r, x);
 }
 return -1;
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

3. Which algorithm is shown below?

```
static int func(int x)
{
 int lo = 0, hi = (arr.length - 1);
 while (lo <= hi && x >= arr[lo] && x <= arr[hi])
 {
 if (lo == hi)
 {
 if (arr[lo] == x) return lo;
 return -1;
 }
 int pos = lo + (((hi-lo) /
 (arr[hi]-arr[lo]))*(x - arr[lo]));
 if (arr[pos] == x)
 return pos;
 if (arr[pos] < x)
 lo = pos + 1;
 else
 hi = pos - 1;
 }
}
```

```
 }
 return -1;
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

4. Which algorithm is shown below?

```
void func(int arr[])
{
 int n = arr.length;

 for (int i = 0; i < n-1; i++)
 {
 int min_idx = i;
 for (int j = i+1; j < n; j++)
 if (arr[j] < arr[min_idx])
 min_idx = j;
 int temp = arr[min_idx];
 arr[min_idx] = arr[i];
 arr[i] = temp;
 }
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

5. Which algorithm is shown below?

```
void func(int arr[])
{
 int n = arr.length;
 for (int i = 0; i < n-1; i++)
 for (int j = 0; j < n-i-1; j++)
 if (arr[j] > arr[j+1])
 {
 int temp = arr[j];
 arr[j] = arr[j+1];
 arr[j+1] = temp;
 }
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

6. Which algorithm is shown below?

```
void func1(int arr[], int l, int m, int r)
{
 int n1 = m - l + 1;
 int n2 = r - m;
 int L[] = new int[n1];
 int R[] = new int[n2];
 for (int i = 0; i < n1; ++i)
 L[i] = arr[l + i];
 for (int j = 0; j < n2; ++j)
 R[j] = arr[m + 1 + j];
 int i = 0, j = 0;
 int k = l;
 while (i < n1 && j < n2) {
 if (L[i] <= R[j]) {
 arr[k] = L[i];
 i++;
 }
 else {
 arr[k] = R[j];
 j++;
 }
 k++;
 }

 while (i < n1) {
 arr[k] = L[i];
 i++;
 k++;
 }
 while (j < n2) {
 arr[k] = R[j];
 j++;
 k++;
 }
}

void func2(int arr[], int l, int r)
{
 if (l < r) {
 int m = (l + r) / 2;
 func2(arr, l, m);
 func2(arr, m + 1, r);
 func1(arr, l, m, r);
 }
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

7. Which algorithm is shown below?

```
int func1(int arr[], int low, int high)
{
 int pivot = arr[high];
```

```
int i = (low-1);
for (int j=low; j<high; j++)
{
 if (arr[j] < pivot)
 {
 i++;
 int temp = arr[i];
 arr[i] = arr[j];
 arr[j] = temp;
 }
}

int temp = arr[i+1];
arr[i+1] = arr[high];
arr[high] = temp;
return i+1;
}
void func2(int arr[], int low, int high)
{
 if (low < high)
 {
 int pi = func1(arr, low, high);
 func2(arr, low, pi-1);
 func2(arr, pi+1, high);
 }
}
```

- a) linear search
- b) binary search
- c) interpolation search
- d) selection sort
- e) bubble sort
- f) merge sort
- g) quick sort

### CS 3173: Basic Computer Architecture

#### Collaborate as a member of a team (introduced)

Do some class activities as groups. For instance, breakout rooms in Zoom or discussing a list of potential test questions in a group. At the end of the semester, include the following questions as part of final exam. Students will score themselves (self-evaluation). Answers will not be scored for correctness, but they will need to answer all questions to get the points (10% of final exam score).

#### Work-Related Interactions with Others (self-evaluation)

Score yourself on your participation in the breakout rooms/ group discussions during the course.  
There are no right or wrong answers, but you do have to answer all questions.

- a. Collaboration: How well did you cooperatively work with others? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- b. Participation: How well did you contribute your “fair share” to the tasks? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- c. Attitude1: How positive was your approach doing the tasks? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- d. Attitude2: How well did you make constructive comments? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- e. Communication: How clearly did you express your thoughts? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)

- f. Responsiveness: How sensitively did you respond to verbal and nonverbal cues of other team members? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- Future (essay): What single thing will you try to do better in the future when working in groups?

**Explain core elements of computer science theory (reinforced)**

1. If a digital thermometer measures between - 100 and +300 degrees, and the digital readout is 0110 1100 (8 bit), what is the analog value being measured?
  - a) 127.11
  - b) 84.71
  - c) 69.41
  - d) 172.95
2. What is the hexadecimal value for a binary unsigned integer of 0011 1001?
  - a) 57
  - b) 39
  - c) 1100 0111
  - d) does not exist
3. What is the binary two's complement for 1011 1011?
  - a) 0100 0101
  - b) 0100 0100
  - c) BB
  - d) does not exist
4. The binary two's complement 1011 0111 has a decimal value of:
  - a) 183
  - b) 73
  - c) -37
  - d) -73
5. An 8-bit value has a minimum of -1,023 and a maximum of + 1,023. The number is a(n)
  - a) binary coded decimal
  - b) unsigned integer
  - c) two's complement
  - d) signed integer
6. The 32-bit binary float of 1100 1000 0101 1111 0101 1100 1001 0000 has a decimal value of:
  - a)  $-1.8 \times 2^{-17}$
  - b)  $+1.8 \times 2^{-17}$
  - c)  $-1.8 \times 2^{17}$
  - d)  $+1.8 \times 2^{17}$
7. The subnet for the IP address 108.235.251.35/22 ranges from:
  - a) 108.235.251.1 - 108.235.251.254
  - b) 108.235.248.1 - 108.235.251.254
  - c) 108.235.251.1 - 108.235.252.254
  - d) 108.235.251.1 - 108.235.254.254
8. The datastream 10110101000 is converted to a Hamming code and arrives as 001101000101000. Which bit of the original data stream was corrupted?
  - a) the sixth bit
  - b) the fifth bit
  - c) the fourth bit
  - d) there was no corruption
9. What is the chip select for a memory device with a low hexadecimal address of D8000 and a high hexadecimal address of DBFFF?
  - a) 11011011

- b) 1101
- c) 1101101
- d) 110110

52. What best describes the difference between Dynamic RAM (DRAM) and Static RAM (SRAM)?
- a) SRAM is faster and cheaper
  - b) SRAM is faster and DRAM is cheaper
  - c) DRAM is faster and SRAM is cheaper
  - d) DRAM is faster and cheaper

### **CS 3343 Computer Operating Systems**

#### **Explain core elements of computer science theory (mastered)**

1. \_\_\_\_\_ involves moving all or part of a process from main memory to disk.
  - a) Swapping
  - b) Relocating
  - c) Suspending
  - d) Blocking
2. The portion of the operating system that selects the next process to run is called the \_\_\_\_\_.
  - a) program status word
  - b) trace
  - c) process control block
  - d) dispatcher
3. A situation in which a runnable process is overlooked indefinitely by the scheduler, although it can proceed, is \_\_\_\_\_.
  - a) mutual exclusion
  - b) starvation
  - c) deadlock
  - d) livelock
4. The requirement that when one process is in a critical section that accesses shared resources, no other process may be in a critical section that accesses any of those shared resources is \_\_\_\_\_.
  - a) atomic operation
  - b) critical section
  - c) livelock
  - d) mutual exclusion
5. A \_\_\_\_\_ is an integer value used for signaling among processes.
  - a) message
  - b) mutex
  - c) semaphore
  - d) atomic operation
6. The \_\_\_\_\_ condition can be prevented by requiring that a process request all its required resources at one time and blocking the process until all requests can be granted simultaneously.
  - a) no preemption
  - b) mutual exclusion
  - c) circular wait
  - d) hold and wait
7. A \_\_\_\_\_ is issued if a desired page is not in main memory.
  - a) page fault
  - b) paging error

- c) page re-placement policy
  - d) page placement policy
8. Which of the following scheduling policies allow the O/S to interrupt the currently running process and move it to the Ready state?
- a) Non-preemptive
  - b) Preemptive
  - c) First In First Out
  - d) First Come First Serve
9. \_\_\_\_\_ is a technique that smoothes out peaks in I/O demand.
- a) Buffering
  - b) Blocking
  - c) Smoothing
  - d) Tracking
10. The most common form of file structure is \_\_\_\_\_
- a) the pile
  - b) the indexed file
  - c) the sequential file
  - d) the indexed sequential file

### **CS 3403: Data Structures**

#### **Apply computer algorithms to solve problems (mastered)**

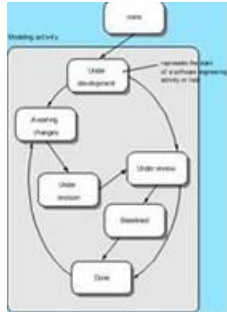
1. Which of the below given sorting techniques has the highest best-case runtime complexity?
  - a) Quick Sort
  - b) Selection Sort
  - c) Insertion Sort
  - d) Bubble Sort
2. The time required to merge two sorted lists of size m and n, is
  - a)  $O(m + n)$
  - b)  $O(m | n)$
  - c)  $O(m \log n)$
  - d)  $O(n \log m)$
3. What is the worst case time complexity of the linear search algorithm?
  - a)  $O(n^2)$
  - b)  $O(1)$
  - c)  $O(\log n)$
  - d)  $O(n)$
4. Which of the following searching techniques do not require the data to be in sorted form?
  - a) Interpolation Search
  - b) Binary Search
  - c) Linear Search
  - d) All the other answers
5. On which principle does the stack work?
  - a) Shortest Process Next
  - b) First Come First Serve
  - c) First In First Out
  - d) First In Last Out
6. In AVL trees, the fastest operation is \_\_\_\_\_.

- a) insertion
  - b) deletion
  - c) retrieval
  - d) updating
7. Which of the following linked lists below have the last node of the list pointing to the first node?
- a) circular linked list
  - b) circular doubly linked list
  - c) circular singly linked list
  - d) doubly linked list
8. In a heap, the element with the greatest key is always in the \_\_\_\_\_
- a) first node of right sub tree
  - b) leaf node
  - c) first node of left sub tree
  - d) root node
9. Which of the following ways is a pre-order traversal?
- a) Root->left sub tree-> right sub tree
  - b) Root->right sub tree-> left sub tree
  - c) right sub tree-> left sub tree->Root
  - d) left sub tree-> right sub tree->Root
10. A full binary tree with n leaves contains \_\_\_\_\_ nodes.
- a) N
  - b)  $n^2$
  - c)  $\log n^2$
  - d)  $2n - 1$
11. What is the average time complexity of the heap sort?
- a)  $O(n^2)$
  - b)  $O(n \log n)$
  - c)  $O(\log n)$
  - d)  $O(n)$

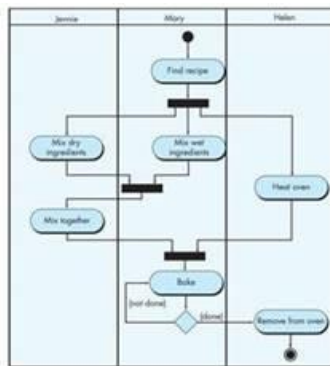
### **CS 4203: Software Engineering**

#### **Create software using software engineering principles (mastered)**

1. Does software suffer from "wearing out"? Give the best answer
- a) yes, instructions deteriorate over time
  - b) yes, due to changing requirements
  - c) yes, but you can just reinstall it
  - d) no, but we do have spikes in software failures when new features are introduced
2. How is most software built?
- a) custom-built
  - b) constructed from components
  - c) by combining different software packages
  - d) from bits and bytes

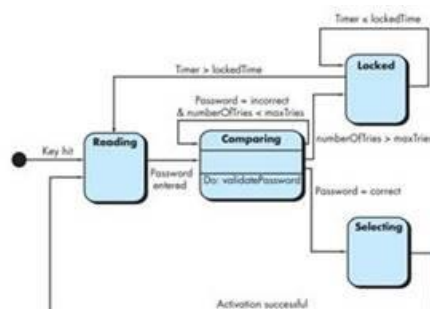


3. This picture shows the \_\_\_\_\_ (model)
- waterfall model
  - incremental model
  - spiral model
  - concurrent model



4. This picture shows a(n) \_\_\_\_\_ (diagram)
- activity diagram
  - activity diagram with swim lanes
  - state diagram
  - use-case diagram

5. What is the most important rule for use case diagrams?
- for the use case circle, use a verb and noun(s)
  - only people are external to the system
  - each use case is connected to at least two external entities
  - each external entity is connected to at least two use cases



6. Which type of diagram is shown in this picture?
- state diagram
  - activity diagram
  - class diagram
  - sequence diagram

7. What are the two major ways to PERMANENTLY store data?
  - a) flat files and databases
  - b) variables and objects
  - c) arrays and lists
  - d) CDROM and DVD
8. If you refactor code, and you try to reduce redundancy, how can you do that?
  - a) creating functions that can be called repeatedly from multiple places
  - b) use global variables
  - c) use constants
  - d) include more libraries
9. If you refactor code, and you want to remove unused design elements, how can you typically spot them?
  - a) they are underlined with a green squiggly line
  - b) they are underlined with a red squiggly line
  - c) debugging stops and the line with the unused element is highlighted red
  - d) use Debug / Refactoring / Redundancy
10. In cohesion and coupling, we want .....
  - a) tight cohesion and loose coupling
  - b) tight cohesion and tight coupling
  - c) loose cohesion and loose coupling
  - d) loose cohesion and tight coupling

#### **Collaborate as a member of a team (mastered)**

Do some class activities as groups. For instance, breakout rooms in Zoom or discussing a list of potential questions on the test. At the end of the semester, I include the following questions as part of final exam. Students will score themselves (self-evaluation). Answers will not be scored for correctness, but they will need to answer all questions to get the points (10% of final exam score).

#### Work-Related Interactions with Others (self-evaluation)

Score yourself on your participation in the breakout rooms/ group discussions during the course.

There are no right or wrong answers, but you do have to answer all questions.

- a. Collaboration: How well did you cooperatively work with others? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- b. Participation: How well did you contribute your “fair share” to the tasks? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- c. Attitude1: How positive was your approach doing the tasks? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- d. Attitude2: How well did you make constructive comments? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- e. Communication: How clearly did you express your thoughts? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- f. Responsiveness: How sensitively did you respond to verbal and nonverbal cues of other team members? (1 poor – 2 fair – 3 average – 4 good – 5 excellent)
- Future (essay): What single thing will you try to do better in the future when working in groups?

#### **Solve problems by applying the software development process (reinforced)**

1. The best strategy to follow when developing software that consists of many modules is to?
  - a) Test each code module separately then link all modules together and test the software as a whole
  - b) Combine unit and integration testing
  - c) Test the software product only once, at the end of the software development
  - d) All of the above

2. A high level of cohesion is an indication that a module?
  - a) Can be written more compactly
  - b) Focuses on just one function
  - c) Interacts with many modules
  - d) Can perform variety of functions
  
3. Which of the following are your best choice for developing Test cases derived from "Equivalence classes and boundary value analysis." for the following scenario: The daily discount must be in the range between and including \$0.00 and \$100.00.
  - a) Testing the values: \$0.00, \$11.01, \$59.31, \$82.02, \$100.00
  - b) Testing the values: -\$1.00, \$0.00, \$1.00, \$50.00, \$99.00, \$100.00, \$101.00
  - c) Testing the values: \$0.00, \$11.21, \$21.11, 28.82, \$82.28, \$100.00, \$000.01
  - d) Random Testing of at least 10% of the values in the range between and including \$0.00 and \$100.00
  
4. All else being equal, which is more desirable?
  - a) Higher cohesion and higher coupling
  - b) Higher cohesion and lower coupling
  - c) Lower cohesion and lower coupling
  - d) Lower cohesion and higher coupling
  
5. More time and money are spent in the \_\_\_\_\_ phase than any other phase.
  - a) Requirements
  - b) Analysis
  - c) Maintenance
  - d) Implementation

### CS 4233: Professional Development in CS

#### Communicate technical ideas both orally and in writing (mastered)

|                 | <b>1(Unacceptable)</b>                                                                                                                                                 | <b>2 (Marginal)</b>                                                                                                         | <b>3 (Good)</b>                                          | <b>4 (Excellent)</b>                                                                                                                                                                               | <b>Score</b> |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| Body Language   | No movement or descriptive gestures.                                                                                                                                   | Very little movement or descriptive gestures.                                                                               | Movements or gestures enhance articulation.              | Movements seemed fluid and helped the audience visualize.                                                                                                                                          |              |
| Eye Contact     | No eye contact with audience.                                                                                                                                          | Minimal eye contact with audience.                                                                                          | Consistent use of direct eye contact with some audience. | Holds attention of entire audience with the use of direct eye contact.                                                                                                                             |              |
| Speaking Skills | <ul style="list-style-type: none"> <li>▪ inaudible or too loud</li> <li>▪ rate too slow/fast</li> <li>▪ speaker seemed uninterested and used monotone voice</li> </ul> | <ul style="list-style-type: none"> <li>▪ some mumbling</li> <li>▪ uneven rate</li> <li>▪ little or no expression</li> </ul> | Clear articulation but not as polished                   | <ul style="list-style-type: none"> <li>▪ Poised, clear articulation</li> <li>▪ proper volume</li> <li>▪ steady rate</li> <li>▪ good posture</li> <li>▪ enthusiasm</li> <li>▪ confidence</li> </ul> |              |
| Organization    | ▪ displays neither clear                                                                                                                                               | ▪ displays some level of                                                                                                    | ▪ displays introductory or closing                       | ▪ delivers clear opening and closing                                                                                                                                                               |              |

|          |                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                |  |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
|          | <p>introductory nor closing remarks</p> <ul style="list-style-type: none"> <li>▪ does not present the segments of the body of the presentation in a coherent manner</li> <li>▪ irrelevant statements are made</li> <li>▪ leaves the audience wondering where the presentation is headed.</li> </ul> | <p>organization with discernible theme, but the presentation is not organized clearly or in a coherent manner.</p> <ul style="list-style-type: none"> <li>▪ introductory and closing remarks are missing.</li> </ul> | <p>remarks, but segments of the body of the presentation are not presented in a coherent manner.</p> <ul style="list-style-type: none"> <li>▪ presents the segments of the body of the presentation in a coherent manner, but introductory or closing remarks are missing.</li> </ul> | <p>remarks that capture the attention of the audience and set the mood</p> <ul style="list-style-type: none"> <li>▪ provides a “road map” for the audience</li> <li>▪ each segment relates to the others according to a carefully planned framework</li> </ul> |  |
| Voice    | Consistently uses a monotone voice                                                                                                                                                                                                                                                                  | Displays some level of inflection throughout delivery.                                                                                                                                                               | Satisfactory use of inflection, but does not consistently use fluid speech.                                                                                                                                                                                                           | Use of fluid speech and inflection maintains the interest of the audience.                                                                                                                                                                                     |  |
| Visuals  | Used no visuals.                                                                                                                                                                                                                                                                                    | <ul style="list-style-type: none"> <li>▪ ran too quickly through visuals and spoke more to the screen than to the audience</li> <li>▪ visuals did not detract from the presentation.</li> </ul>                      | <ul style="list-style-type: none"> <li>▪ gave audience almost enough time to absorb material, but occasionally read the slide</li> <li>▪ visuals added to the presentation.</li> </ul>                                                                                                | <ul style="list-style-type: none"> <li>▪ gave audience ample time to absorb information on visual</li> <li>▪ spoke to the audience, not the screen</li> <li>▪ visuals greatly enhanced presentation.</li> </ul>                                                |  |
| Language | Multiple grammar errors and use of inappropriate vocabulary.                                                                                                                                                                                                                                        | <ul style="list-style-type: none"> <li>▪ one or two minor grammar errors.</li> <li>▪ vocabulary use is too elementary or not effective</li> </ul>                                                                    | <ul style="list-style-type: none"> <li>▪ correct grammar</li> <li>▪ vocabulary mostly appropriate for the purpose and the audience</li> </ul>                                                                                                                                         | <ul style="list-style-type: none"> <li>▪ correct use of grammar</li> <li>▪ use of some advanced language</li> <li>▪ effective use of appropriate vocabulary for the purpose and for the audience</li> </ul>                                                    |  |
| Totals   |                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                       |                                                                                                                                                                                                                                                                |  |

**Solve problems by applying the software development process (mastered)**

Student(s): \_\_\_\_\_

Capstone Assessment

3 = Good    2 = Average    1 = Below Average    0 = Not gradable

| Criteria                                                                                  | Comments | Points |   |   |   |
|-------------------------------------------------------------------------------------------|----------|--------|---|---|---|
| The project includes a clearly defined problem or need                                    |          | 3      | 2 | 1 | 0 |
| The student provided sufficiently frequent status updates                                 |          | 3      | 2 | 1 | 0 |
| The project includes modeling diagrams that accurately describe the context               |          | 3      | 2 | 1 | 0 |
| All project files and source code required to build and execute are submitted             |          | 3      | 2 | 1 | 0 |
| The software runs without error, and has reasonable error checking and exception handling |          | 3      | 2 | 1 | 0 |
| The interfaces have a professional appearance                                             |          | 3      | 2 | 1 | 0 |
| The project includes a comprehensive test plan                                            |          | 3      | 2 | 1 | 0 |
| The project includes documentation for installation and operation                         |          | 3      | 2 | 1 | 0 |
| Total Points out of 24:                                                                   |          |        |   |   |   |
| Comments:                                                                                 |          |        |   |   |   |

**CS 4343: Database Management Systems**

**Create software using software engineering principles (reinforced)**

- One disadvantage of spreadsheet in comparison to database management systems is:
  - reduced data duplication.
  - program-data independence.
  - limited data sharing.
  - enforcement of integrity constraints.
- A join that keeps unmatched rows from one or both tables is called a(n):
  - equi-join.
  - outer join.
  - multivariate join.
  - inner join.
- List the types of database applications. (Please feel free to choose more than one correct answer.)
  - PC databases
  - workgroup databases

- c) dept databases
  - d) enterprise databases
  - e) all of above
4. In term of relational database, Conceptual schema includes:
- a) Conceptual design
  - b) Description of data requirements
  - c) Includes detailed descriptions of the entity types, relationships, and constraints
  - d) Transformed from high-level data model into implementation data model
  - e) All of above
5. Which of the following is not a component of ER models?
- a) Entities
  - b) Relationships
  - c) Attributes
  - d) Store procedures
6. The CARTESIAN PRODUCT (Cross product) operation: Choose all that are true.
- a) useful when followed by a selection that matches values of attributes
  - b) denoted by x
  - c) unary set operation
  - d) relations do not have to be union compatible
  - e) A, B, D
7. Match the following with each statement
- Data definition language (DDL)  
Storage Definition language (SDL)  
View Definition Language (VDL)  
Data Manipulation Language (DML)
- a) allows retrieval, insertion, deletion, modification
  - b) specifies the internal schemas
  - c) defines both schemas
  - d) Specifies user views/mappings to conceptual schema

### **MATH 3023: Discrete Mathematics**

1. Add the following binary numbers
- $$\begin{array}{r} 10101111 \\ + 11000011 \\ \hline \end{array}$$
- a) 101110010
  - b) 01110010
  - c) 111110010
  - d) None of these answers is correct
2. Add the following hexadecimal numbers
- $$\begin{array}{r} ABC \\ + DEF \\ \hline \end{array}$$
- a) 18AB
  - b) 24AB
  - c) 181B
  - d) 182B
3. Add the following hexadecimal numbers
- $$\begin{array}{r} BAA \\ + AAA \\ \hline \end{array}$$

- a) 1FFD
  - b) 16FE
  - c) 1120
  - d) 1654
4. Given that  $p$  and  $q$  are false and  $r$  is true, evaluate the proposition  $(q \rightarrow r) \rightarrow (\neg r \rightarrow p)$
- a) FALSE
  - b) TRUE
  - c) UNKNOWN
  - d) None of these answers is correct
5. Given that  $p$  and  $q$  are false and  $r$  is true, evaluate the proposition  $(r \leftrightarrow p) \vee (p \rightarrow q)$
- a) FALSE
  - b) TRUE
  - c) UNKNOWN
  - d) None of these answers is correct
6. In how many ways can we select a committee of 3 women and 3 men from a group of 5 distinct women and 7 distinct men?
- a) 45
  - b) 350
  - c) 140
  - d) None of these answers is correct
7. How many strings can be formed using the letters  $M I S S I S S I P P I$ ?
- a) 34650
  - b) 11!
  - c) 11
  - d) None of these answers is correct
8. Given the sets  $A = \{1, 3, 5, 9\}$ ,  $B = \{1, 2, 3, 5, 8\}$ ,  $C = \{2, 4, 6, 10\}$ ,  $U = \{1, 2, 3, \dots, 10\}$ , list the elements that result from the following set operation:  $(B - C) - A$
- a)  $\{9\}$
  - b)  $\{1, 3, 5, 8\}$
  - c) The empty set
  - d) None of these answers is correct

### Appendix E – Programming Assignments

Programmer(s)

Program Assessment  
3 = Good    2 = Average    1 = Below Average    0 = Not gradable

| Criteria                                                                                                | Comments | Points |   |   |   |
|---------------------------------------------------------------------------------------------------------|----------|--------|---|---|---|
| 1. The program fully implements the solution to the given problem                                       |          | 3      | 2 | 1 | 0 |
| 2. The solution produces the correct result                                                             |          | 3      | 2 | 1 | 0 |
| 3. The program runs without error, has reasonable error checking and exception handling                 |          | 3      | 2 | 1 | 0 |
| 4. The user is prompted for input when required                                                         |          | 3      | 2 | 1 | 0 |
| 5. The user is presented with appropriate feedback as to the results of an operation                    |          | 3      | 2 | 1 | 0 |
| 6. All project files and source code required to build and execute are submitted                        |          | 3      | 2 | 1 | 0 |
| 7. Code is properly indented, and adequately commented                                                  |          | 3      | 2 | 1 | 0 |
| 8. Code is well written and free from unnecessary complexity or redundancy                              |          | 3      | 2 | 1 | 0 |
| 9. Programmer defined symbol names are reasonable (variable names, class names, function names, etc...) |          | 3      | 2 | 1 | 0 |
| 10. The user interface is accurate and is free from misspelled words and bad grammar                    |          | 3      | 2 | 1 | 0 |
| Total Points out of 30:                                                                                 |          |        |   |   |   |
| Comments:                                                                                               |          |        |   |   |   |