ISSN: 2167-1435
v31 n3002

IT0: Discrete Math and Programming Logic Topics as a Hybrid Alternative to CS0

Nancy L. Martin
nlmartin@siu.edu
Information Systems Technologies
Southern Illinois University
Carbondale, IL 62901, U.S.A.

Abstract

This paper describes the development of a hybrid introductory course for students in their first or second year of an information systems technologies degree program at a large Midwestern university. The course combines topics from discrete mathematics and programming logic and design, a unique twist on most introductory courses. The objective of the new course is to better prepare students for more advanced computing courses. Two primary drivers motivated development of the new course: 1) faculty evidence of deficient foundation skills in advanced level courses, and 2) consideration of program accreditation criteria.

Keywords: Course Design, Introductory Course, Programming, Discrete Mathematics, IT2008 Model Curriculum, Course Development

1. PROGRAM BACKGROUND

Since its inception more than twenty years ago, the Information Systems Technologies (IST) degree program has continually evolved to meet the changing needs of its stakeholders. The IST program is housed in the College of Applied Sciences and Arts, reflecting its focus on applied, hands-on skills in the field of information technology (IT). Since the late 1990's, the IST major has progressed from an office systems degree that was based on the Organizational and End-user Information Systems (OEIS) curricula guide (The Organizational Systems Research Association, 2004) to today's program with courses in programming, networking, databases, web systems development, and other core topics. Concentrations are currently available in two tracks: network and information security and web systems development.

As the IST major grew from the OEIS model, more and more technical courses were added to the curriculum. For example, several information assurance-focused courses are now part of the Network and Information Security Track, and the program earned designation as a National Center of Academic Excellence in Information Assurance Education in 2011. In recent years, faculty have implemented numerous new elective courses and revised courses in the core and tracks to keep the curriculum current with the needs of the program stakeholders and the job market. Also, as the program has progressed, the technical components of the courses have become more rigorous.

The IST program consists of 16 core courses plus elective courses, some of which are offered in specialized tracks. Students may complete a track or combine a variety of elective courses to create a more personalized program. University core curriculum requirements comprise 41 credit hours; the IST core comprises another 49 credit hours, and the remaining 30 hours are open to IST electives. All courses are one-semester, three credit hours except the required internship which is four credit hours. For a mathematical foundation, IST students have been required to take an applied statistics course and only the

minimal mathematics required in the university core curriculum. Students are encouraged to take a philosophy-based logic course as part of the university core curriculum, but it is not required. The IST core courses, electives, and tracks are listed in Appendix A.

Student Demographics

In recent years, average enrollment for the IST program has been 215 students, and the program confers an average of 73 degrees annually. Most students are male, with only about 15 percent female enrollment. About five percent of students enroll directly into the program as freshman. The majority of students transfer from pre-major or other majors within the university, and roughly 11 percent transfer from community colleges or other universities. Most within-university transfers come from computer science and computer engineering. The IST program does not require calculus or other upper level mathematics courses which draws many students to the program from computer science and engineering. The retention rate in IST is one of the highest in the university, averaging over 90 percent.

IST graduates are recruited by a number of international, national, and regional companies. Examples of major employers include a large aircraft company, a national insurance company, an international information security company, and a national healthcare software vendor. A recent survey of graduates found that about 32 percent were employed or had job offers prior to graduation and another 53 percent were employed within six months of graduation (Legier, Woodward, & Martin, 2013).

2. MOTIVATION

Two primary factors motivated the development of a new foundation course in the IST major. First, anecdotal evidence from faculty revealed that some students in higher level courses struggle with concepts normally covered in prerequisite courses such as discrete mathematics and programming logic. Second, faculty began considering the feasibility of seeking accreditation for the IST program.

Student Performance

Curriculum enhancements over the past several years have created a stronger, more current program with courses such as advanced web systems development, software engineering, and advanced enterprise networking. As more advanced courses were developed, lower level courses were also updated to better equip students both for later courses and also for the job market in general. For example, systems analysis and design was previously taught as a two course sequence with the first course offered in the sophomore year and the second course offered in the senior year. Those two courses have now been combined into one upper level course. Another example is that client side web technologies was taught as a separate course, but that content has now been "pushed" down into the introductory web applications course. Additionally, students were only required to take one Java-based programming course, and relatively few took the Programming II course. Coverage in the first course was restricted to basic programming concepts and initial coverage of arrays, which is considered limited programming knowledge in the IST curriculum.

Over time it became clear to faculty of upper level programming, web development, and network and security courses that some students lacked foundation skills needed to be successful in those advanced topics. For example, some students found their initial real application of binary and hexadecimal number systems in the first information assurance course. The instructor found it necessary to spend valuable course time reviewing those topics. Similarly, faculty in advanced web systems and software engineering courses spent too much reviewing basic programming concepts.

Curriculum Review for Accreditation

Curriculum review is an ongoing process for IST faculty. However, in spring of 2013, IST faculty undertook a comprehensive review of the curriculum in consideration of pursuing program accreditation through the Accreditation Board for Engineering and Technology (ABET). The first step was to determine where the IST program best fit within ABET computing programs. The IST program was obviously not a computer science program, so only information systems and information technology programs were considered. Also, as part of the review process, model curricula were considered for information technology (IT2008) (Lunt et al., 2008) and information systems (IS2010) (Topi et al., 2010).

Careful comparison of IST program objectives and desired student outcomes with ABET

accreditation criteria revealed a good alignment with ABET's information technology program (ABET, 2012). Similarly, the IT2008 curriculum model philosophy, body of knowledge, and learning outcomes most closely fit with the existing IST program. Although the entire review is outside the scope of this paper, two areas were important in the creation of a new foundation course: programming and mathematics.

3. COMPUTING CURRICULA

There are a variety of computing degrees available to today's college students. The Association of Computing Machinery defines five distinct computing curricula (CC2005): computer engineering (CE), computer science (CS), information systems (IS), information technology (IT), and software engineering (SE) (ACM/AIS/IEEE-CS Joint Task Force for Computing Curricula, 2005). The difference among the five is a varying emphasis on computing knowledge areas, goals, and capabilities of graduates. It is also important to note that there are options other than the five distinct areas since some programs blur the lines of distinction between the ACM curricula (e.g., Connolly & Paterson, 2011). Regardless of the curricula followed, one commonality is that some degree of programming knowledge and some level of mathematics are recommended.

CC2005, for the first time, defined IT separately from other computing degrees. Soon after, the first model curriculum, IT2008, was released and provided this definition: "IT, as an academic discipline, is concerned with issues related to advocating for users and meeting their needs within an organizational and societal context through the selection, creation, application, integration and administration of computing technologies" (Lunt, et al., 2008, p. 9). IT degree programs had arisen from an industry need for "professionals to select, create, apply, integrate, and administer an organizational IT infrastructure" (Lunt, Ekstrom, Reichgelt, Bailey, & LeBlanc, 2010, p. 133), and that need was not being met by computer science or information systems programs. The IST program evolved in the same way and in parallel to the IT academic discipline.

Programming in the IT2008 Curriculum

IT2008 defines 13 knowledge areas which are subdivided into units and topics within units. The Programming Fundamentals knowledge area comprises five units with the recommended minimum coverage hours displayed in Table 1. The recommended coverage totals 38 hours, which, in a three-credit hour course environment, could be delivered in a one semester course. This approach is the most common in computing programs. IT2008 points out "that the number of core hours prescribed for this knowledge area is dependent on some previous programming experience" (Lunt, et al., 2008, p. 103).

Unit	Recommended Min Hours
Fundamental Data Structures	10
Fundamental Programming Constructs	10
Object-Oriented Programming	9
Algorithms and Problem-Solving	6
Event-Driven Programming	3

Table 1. IT2008 Programming Fundamentals Units

The introductory programming course, often called CS1, is the cornerstone of any computing curricula. However, over the past decade, universities have become increasingly concerned about declining enrollments and retention in computing programs. Eager to find solutions to those problems, the CS1 course has been an obvious place to focus efforts. Many students come into a computing major with little or no previous programming experience, a category of learners dubbed "novice programmers". If student performance in CS1 can be improved, future success in a computing major is more likely.

However, learning to program is notoriously difficult for novice programmers, and as a result, much attention and research has focused on this perennial problem. (See Robins, Rountree, & Rountree, 2003). Many have studied the characteristics, habits, and success factors of novices in the CS1 course (e.g., Bennedsen & Caspersen, 2005a; Porter, Guzdial, McDowell, & Simon, 2013; Rountree, Rountree, & Robins, 2002). Others have focused on specific methods or approaches to improve performance in the CS1 course (e.g., Benander & Benander, 2008; Bennedsen & Caspersen, 2005b; Gill & Holton, 2006; Pears et al., 2007; Vihavainen, Paksula, & Luukkainen, 2011; Williams, Wiebe, Yang, Ferzli, & Miller, 2002; Zhang, Zhang, Stafford, & Zhang, 2013).

The CS0 Course

One approach to improve CS1 performance is to require CS0, a "preprogramming" course. In some computing curricula, the CS1 course is preceded by CS0 or some other form of introductory course, and these prerequisite courses can improve performance in CS1 (Brown, 2013; Chor & Hod, 2012; Dierbach, Taylor, Zhou, & Zimand, 2005; Faux, 2006).

CS0 was first introduced as an orientation to the computer science major (Cook, 1997), and was implemented similarly to orientation courses in other disciplines. In addition to basic computing skills, the course included topics on time management, problem solving, professionalism, and career exploration. Over time, CS0 course designers embraced a variety of approaches and topics. Two common formats are breadth-first and depth-first. In a breadth-first CS0 course, exposure to programming language is limited to basic concepts. Topics may include those similar to (Cook, 1997), or focus on "authentic" everyday computing tasks to help students more easily comprehend computer science concepts (McFall & DeJongh, 2011). A depth-first approach usually depends on a specific programming language to develop problem-solving skills (Tucker & Garnick, 1991). More recently, CS0 courses are being implemented using a high level language in an attempt to attract students to a major in computing or to improve retention rates for at-risk students (e.g., Rizvi & Humphries, 2012; Uludag, Karakus, & Turner, 2011).

Another common form of CS0 is a programming logic course; however, some have found that particular type of course did not improve students' performance in advanced programming (Hoskey & Murino, 2011). Others have developed the CS0 course to address specific deficiencies. For example, one CS0 course focuses on mental models and concepts of programming (Dierbach, et al., 2005). Another found that emphasis on problem solving techniques and algorithm development prior to programming is beneficial (Faux, 2006). Others have also reported success with a CS0 course focused on problem solving skills (e.g., Cortina, 2007; Middleton, 2012; Mitchell, 2001; Van Dyne & Braun, 2014).

Regardless of the course arrangement or focus, the common objective of a CS0 course remains: to improve success in subsequent programming and computing courses. With the same objective in mind and recalling that the IT2008 coverage recommendations for programming assume some sort of previous exposure or experience, the IST faculty began planning the development of an introductory course, IT0.

Mathematics in the IT2008 Curriculum

IT2008 also addresses fundamental IT knowledge areas including mathematical foundations. The Math and Statistics for IT knowledge area comprises seven units with the specific recommended minimum coverage displayed in Table 2. The total recommended coverage is 38 hours.

Unit	Recommended Min Hours
Basic Logic	10
Discrete Probability	6
Functions, Relations and Sets	6
Hypothesis Testing	5
Sampling and Descriptive Statistics	5
Graphs and Trees	4
Application of Math & Statistics to IT	2

Table 2. IT2008 Math and Statistics for IT Units

The emphasis in IT2008 is on topics in statistics and discrete mathematics with a notable absence of calculus. Rigorous math requirements such as calculus have likely scared more than a few students away from computing majors, especially since some students have difficulty understanding how abstract mathematical concepts relate to the real world.

In research, the relationship between students' math background and success in computing courses has been a topic of interest for decades. How students perform in mathematics courses can sometimes predict success or failure in programming courses or in the entire computing curriculum (e.g., Campbell & McCabe, 1984; Capstick, Gordon, & Salvadori, 1975; Konvalina, Wileman, & Stephens, 1983; White & Sivitanides, 2003; Wilson & Shrock, 2001). Studies have usually focused on the number or type of mathematics courses taken or on scores on standardized tests. Regardless of findings, educators overwhelmingly agree that skills in mathematical thinking and reasoning transfer to success in working with abstract concepts and symbol manipulation in programming (e.g., Bruce, Scot, Kelemen, & Tucker, 2003; Henderson, 2005; Kelemen, Tucker, Henderson, Astrachan, & Bruce, 2000; Ralston, 2005).

ISSN: 2167-1435
v31 n3002

There is also an ongoing debate as to exactly what kind of mathematics is really needed in various computing programs (e.g., Bruce, et al., 2003; Glass, 2000). However, a common theme throughout the literature is that of all mathematics courses taken, discrete mathematics may be the most important predictor of success in computing (Pioro, 2006; Sidbury, 1986). Moreover, a further dissection of the "how much math" debate reveals that most educators agree that topics in discrete mathematics are the most relevant for computing professionals. This sentiment toward coverage of topics in discrete mathematics is evident in the IT2008 model curriculum.

The Discrete Mathematics Course

Discrete mathematics is commonly taught in a one or two semester sequence in computing programs. Topics covered in these courses include logic, sets, functions, relations, counting, proofs, probability, and trees and graphs, among others. As with other mathematics courses, students do not necessarily recognize how discrete mathematics applies to their profession or to their future studies (Remshagen, 2010).

While the approaches to teaching discrete mathematics are not as varied or numerous as those for CS0, some universities have taken an integrative approach, incorporating discrete mathematics topics into core curriculum (Harvey, Wu, Turchek, & Longenecker, 2007), or into other courses such as data structures or formal methods, or simply focusing on making the topics more relevant to students. (Gegg-Harrison, 2005; Remshagen, 2010). Others have argued for combining discrete mathematics and functional programming into one course (VanDrunen, 2011).

Seeing the need for relevance in information systems programs, an interdisciplinary committee of faculty at one university developed a unique discrete mathematics course. The course was designed to relate real world uses and examples to selected discrete mathematics topics, all while "making learning easier and enjoyable" and increasing student confidence (Wood, Harvey, & Kohun, 2005, p. 387). The team developed customized course materials and have found their approach valuable in the information systems curriculum.

4. THE IT0 COURSE

Motivated by the need to better equip IST students for advanced coursework and the possibility of seeking ABET accreditation, the concept for a new IT0 course was formed. While the programming topics recommended in IT2008 were already being covered, the depth of coverage and the assumption of prior experience needed to be addressed. Additionally, within the mathematics knowledge area, some topics were covered in the applied statistics course; however other topics were only being haphazardly addressed elsewhere in the curriculum.

Overall, IST faculty felt that combining portions of both the Math and Statistics and the Programming Fundamentals knowledge areas from IT2008 would create a well-rounded preparatory course for the IST program and would greatly benefit IST students. However, there was not room in the curriculum to incorporate an additional mathematics course and a CS0-type course. Moreover, an entire semester of either course was not deemed necessary for the IST program. An opportunity presented itself when the former two course systems analysis and design sequence was compressed into one course. That change freed up a sophomore level course which would be used to create the hybrid discrete mathematics and programming logic and design course, IT0.

Course Content

With a one course equivalent available in the curriculum, faculty began researching options or models for a combined discrete mathematics topics/programming logic course. Unfortunately, none were found, and it became clear the course would need to be developed from scratch. Since the IST program is most closely aligned with IT2008, faculty turned to those requirements for guidance in creating the new IT0 hybrid course.

The IT2008 Programming Fundamentals knowledge area units and recommended coverage hours were shown in Table 1. All programming units were previously covered in the introductory programming course, however some were covered only at a shallow level due to the time constraint of a one semester course. With the new IT0 course, most units move to the new hybrid course and object-oriented programming will be introduced only in the context of simple problem solving exercises. Coverage for the new course is shown in Table 3.

Unit	Hours	Covered in Intro. Programming Course	Covered in New IT0 Course
Fundamental Data Structures	10	X	
Fundamental Programming Constructs	10	X	X
Object-Oriented Programming	9	X	
Algorithms and Problem-Solving	6	X	X
Event-Driven Programming	3	X	

Table 3. Programming Fundamentals Unit Coverage

IT2008 also provides specific topics and learning outcomes for each knowledge area unit. This detail enabled faculty to clearly define learning objectives for the new course and ensure it met the prerequisite needs of more advanced courses in the IST program. The Programming Fundamentals units and specific topics with the associated learning outcomes as described in IT2008 are provided in Appendix B.

The Math and Statistics knowledge area units and hours were outlined in Table 2. Table 4 displays which of those units are currently covered in the applied statistics course for IST. Upon review, the topics of Basic Logic, Functions, Relations and Sets, and Graphs and Trees were nearly exact matches to the topic list IST faculty had devised as being areas of deficiency. These units will be addressed in the new IT0 course. Specific topics and learning outcomes for the discrete mathematics portion of the new course were also taken from IT2008 and are available in Appendix C.

Unit	Hours	Covered in Statistics Course	Covered in New IT0 Course
Basic Logic	10		X
Discrete Probability	6	X	
Functions, Relations and Sets	6		X
Hypothesis Testing	5	X	
Sampling and Descriptive Statistics	5	X	
Graphs and Trees	4		X
Application of Math & Statistics to IT	2	X	X

Table 4: Math and Statistics for IT Unit Coverage

The master course syllabus identifies the amount of time to be dedicated to each topic area and is included as Appendix D.

Course Materials

Since introductory courses in the IST program are sometimes taught by term instructors, the program requires the use of textbooks and other materials that are listed in the master syllabus for a course as a means to insure consistency. To assure the new IT0 course meets the stated objectives, the next task was to identify standard course materials. Needless to say, one textbook that covered all the IT0 topics did not exist.

Selecting a resource to cover the programming fundamentals portion of the course was fairly easy; a book by the same publisher and author as is used in the introductory programming course was selected. The similarity in the language and approach of the texts would afford a smoother transition to more advanced programming concepts for IST students.

Finding an appropriate resource for the mathematics portion of the class proved to be more difficult. Dozens of books and numerous web resources in the areas of discrete mathematics and introduction to computer science were reviewed. In each case, the text contained a great deal of information that would not be covered and thus not warrant requiring students to purchase a second book for the IT0 course. Faculty then began searching for freely available sources including entire books or individual modules in an effort to provide future instructors with a complete set of course materials. As of this writing, material for the mathematics portion of the IT0 course will be prepared by the assigned instructor.

5. CONCLUSION

The primary goal of creating the IT0 course was to provide IST students with a firm foundation in mathematical reasoning and problem-solving skills while introducing major programming concepts needed for success in more advanced courses. Identifying the IT2008 model curriculum components that closely aligned with the deficiencies observed by the faculty confirmed the need for such a course.

The opportunity to cover basic programming concepts in an earlier course means that the introductory programming course can now

provide deeper coverage of important topics such as object orientation. Also, by providing all students with a solid foundation in discrete mathematics topics and problem-solving skills, higher level courses across the curriculum will benefit. For example, the domino effect of pushing content from the introductory programming course into the IT0 course allows content from the advanced programming course to be pushed into the introductory course. This move allows some courses that previously had Programming II as a prerequisite to now only require introductory programming. Additionally, information security and database programming courses can spend less time covering basic concepts, thereby addressing more advanced content.

The new IT0 course has been approved through university channels and will be offered, and required, for the first time in fall 2014. While the initial work is complete, there is much more to be done. For example, faculty must now measure the effectiveness of the new approach. Plans are underway to provide a pretest and post-test for the IT0 course to ensure learning objectives are being met. Additionally, faculty in upper level courses will monitor the preparedness of IST students, some also using a pretest. It will be difficult to measure the direct impact on the introductory programming course since the content is changing along with the new IT0 prerequisite. However, faculty will be keenly aware of any needed adjustments with the new curriculum.

In addition to measuring effectiveness, future plans for the IT0 course include the development of a custom textbook that will meet the students' needs and serve as a basis in the event instructor assignments change.

Our experience reinforces the fact that each program and its stakeholders are different. While others have found success eliminating a two semester approach to teaching programming (e.g., Colton & Curtis, 2010), ours has been the opposite experience. We believe that the new IT0 course, based on a widely accepted curriculum model, will provide the foundation skills our students need to be successful not only in the IST program, but also as future IT professionals. Further, we hope that our experience in creating a hybrid course to meet specific program needs will be of value to other educators.

6. REFERENCES

ABET. (2012). Criteria for accrediting computing programs. Retrieved from http://www.abet.org/DisplayTemplates/DocsHandbook.aspx?id=3148

ACM/AIS/IEEE-CS Joint Task Force for Computing Curricula. (2005). Computing curricula 2005: The overview report.

Benander, A. C., & Benander, B. A. (2008). Student monks - Teaching recursion in an IS or CS programming course using the Towers of Hanoi. *Journal of Information Systems Education, 19*(4), 455-467.

Bennedsen, J., & Caspersen, M. E. (2005a). *An investigation of potential success factors for an introductory model-driven programming course*. Proceedings of the First International Workshop on Computing Education Research, Seattle, WA.

Bennedsen, J., & Caspersen, M. E. (2005b). Revealing the programming process. *SIGCSE Bulletin, 37*(1), 186-190.

Brown, M. (2013). CS0 as an indicator of student risk for failure to complete a degree in computing. *Journal of Computing Sciences in Colleges, 28*(5), 9-16.

Bruce, K. B., Scot, R. L., Kelemen, C., & Tucker, A. (2003). Why math? *Communications of the ACM, 46*(9), 41-44.

Campbell, P. F., & McCabe, G. P. (1984). Predicting the success of freshmen in a computer science major. *Communications of the ACM, 27*(11), 1108-1113.

Capstick, C. K., Gordon, J. D., & Salvadori, A. (1975). Predicting performance by university students in introductory computing courses. *SIGCSE Bulletin, 7*(3), 21-29.

Chor, B., & Hod, R. (2012). *Cs1001.Py: A topic-based introduction to computer science.* Proceedings of the 17th Annual Conference on Innovation and Technology in Computer Science Education, Haifa, Israel.

Colton, D., & Curtis, A. (2010). Programming proficiency in one semester: Lessons

learned. *Information Systems Education Journal, 8*(48), 3-15.

Connolly, R. W., & Paterson, B. (2011). *Even so with the pieces borrowed from others: Dressing an IS program in IT clothing.* Proceedings of the 2011 Conference on Information Technology Education, West Point, NY.

Cook, C. R. (1997). CS0: Computer science orientation course. *SIGCSE Bulletin, 29*(1), 87-91.

Cortina, T. J. (2007). An introduction to computer science for non-majors using principles of computation. *SIGCSE Bulletin, 39*(1), 218-222.

Dierbach, C., Taylor, B., Zhou, H., & Zimand, I. (2005). Experiences with a CS0 course targeted for CS1 success. *SIGCSE Bulletin, 37*(1), 317-320.

Faux, R. (2006). Impact of preprogramming course curriculum on learning in the first programming course. *IEEE Transactions on Education, 49*(1), 11-15.

Gegg-Harrison, T. S. (2005). Constructing contracts: Making discrete mathematics relevant to beginning programmers. *Journal on Educational Resources in Computing, 5*(2), 3.

Gill, G., & Holton, C. F. (2006). A self-paced introductory programming course. *Journal of Information Technology Education, 5*, 95-105.

Glass, R. L. (2000). A new answer to "How important is mathematics to the software practitioner?". *IEEE Software, 17*(6), 135-136.

Harvey, V. J., Wu, P. Y., Turchek, J. C., & Longenecker, J., Herbert E. (2007). Coordinated topic presentations for information systems core curriculum and discrete mathematics courses. *Information Systems Education Journal, 5*(8), 3-10.

Henderson, P. B. (2005). Mathematics in the curricula. *SIGCSE Bulletin, 37*(2), 20-22.

Hoskey, A., & Murino, P. S. (2011). Beyond introductory programming: Success factors for advanced programming. *Information Systems Education Journal, 9*(5), 61-70.

Kelemen, C., Tucker, A., Henderson, P., Astrachan, O., & Bruce, K. (2000). *Has our curriculum become math-phobic? (an American perspective)*. Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education, Helsinki, Finland.

Konvalina, J., Wileman, S. A., & Stephens, L. J. (1983). Math proficiency: A key to success for computer science students. *Communications of the ACM, 26*(5), 377-382.

Legier, J., Woodward, B., & Martin, N. (2013). Reassessing the skills required of graduates of an information systems program: An updated analysis. *Information Systems Education Journal, 11*(3), 79-89.

Lunt, B., Ekstrom, J. J., Gorka, S., Hislop, G., Kamali, R., Lawson, E., . . . Miller, J. (2008). Information technology 2008, curriculum guidelines for undergraduate programs in information technology: ACM and IEEE Computer Society.

Lunt, B., Ekstrom, J. J., Reichgelt, H., Bailey, M., & LeBlanc, R. (2010). IT 2008: The history of a new computing discipline. *Communications of the ACM, 53*(12), 133-141.

McFall, R. L., & DeJongh, M. (2011). *Increasing engagement and enrollment in breadth-first introductory courses using authentic computing tasks.* Proceedings of the 42nd Technical Symposium on Computer Science Education (pp. 429-434). Dallas, TX.

Middleton, D. (2012). Trying to teach problem-solving instead of just assigning it: Some practical issues. *Journal of Computing Sciences in Colleges, 27*(5), 60-65.

Mitchell, W. (2001). Another look at CS0. *Journal of Computing Sciences in Colleges, 17*(1), 194-205.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., . . . Paterson, J.

ISSN: 2167-1435
v31 n3002

(2007). A survey of literature on the teaching of introductory programming. *SIGCSE Bulletin, 39*(4), 204-223.

Pioro, B. T. (2006). Introductory computer programming: Gender, major, discrete mathematics, and calculus. *Journal of Computing Sciences in Colleges, 21*(5), 123-129.

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in introductory programming: What works? *Communications of the ACM, 56*(8), 34-36.

Ralston, A. (2005). Do we need any mathematics in computer science curricula? *SIGCSE Bulletin, 37*(2), 6-9.

Remshagen, A. (2010). *Making discrete mathematics relevant*. Proceedings of the 48th Annual ACM Southeast Regional Conference, Oxford, MS.

Rizvi, M., & Humphries, T. (2012). *A Scratch-based CS0 course for at-risk computer science majors.* Proceedings of the Frontiers in Education Conference, Seattle, WA.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education, 13*(2), 137-172.

Rountree, N., Rountree, J., & Robins, A. (2002). Predictors of success and failure in a CS1 course. *SIGCSE Bulletin, 34*(4), 121-124.

Sidbury, J. R. (1986). A statistical analysis of the effect of discrete mathematics on the performance of computer science majors in beginning computing classes. *SIGCSE Bulletin, 18*(1), 134-137.

The Organizational Systems Research Association. (2004). Organizational & end-user information systems model curriculum.

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K. M., Nunamaker, Jr., J.F., Sipior, J. C., & de Vreede, G. J. (2010). IS2010: Curriculum guidelines for undergraduate degree programs in information systems.

Tucker, A., & Garnick, D. (1991). A breadth-first introductory curriculum in computing. *Computer Science Education, 2*(3), 271.

Uludag, S., Karakus, M., & Turner, S. W. (2011). *Implementing IT0/CS0 with Scratch, App Inventor for Android, and Lego Mindstorms.* Proceedings of the 2011 Conference on Information Technology Education, West Point, NY.

Van Dyne, M., & Braun, J. (2014). *Effectiveness of a computational thinking (CS0) course on student analytical skills.* Proceedings of the 45th Technical Symposium on Computer Science Education, Atlanta, GA.

VanDrunen, T. (2011). *The case for teaching functional programming in discrete math*. Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications, Portland, OR.

Vihavainen, A., Paksula, M., & Luukkainen, M. (2011). *Extreme apprenticeship method in teaching programming for beginners.* Proceedings of the 42nd Technical Symposium on Computer Science Education, Dallas, TX.

White, G., & Sivitanides, M. (2003). An empirical investigation of the relationship between success in mathematics and visual programming courses. *Journal of Information Systems Education, 14*(4), 409-416.

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In support of pair programming in the introductory computer science course. *Computer Science Education, 12*(3), 197.

Wilson, B. C., & Shrock, S. (2001). Contributing to success in an introductory computer science course: A study of twelve factors. *SIGCSE Bulletin, 33*(1), 184-188.

Wood, D. F., Harvey, V. J., & Kohun, F. (2005). Life-long learning - Making discrete math relevant for information systems professionals. *Issues In Information Systems, VI*(1), 386-390.

ISSN: 2167-1435
v31 n3002

Zhang, X., Zhang, C., Stafford, T. F., & Zhang, P. (2013). Teaching introductory programming to IS students: The impact of teaching approaches on learning performance. *Journal of Information Systems Education, 24*(2), 147-155.

ISSN: 2167-1435
v31 n3002

APPENDIX A
IST Core Courses

Year 1	
	Installing & Upgrading Computer Systems
	Optimizing & Troubleshooting Operating Systems
	Computing for Business Administration
Year 2	
	LAN Installation & Administration
	Fiscal Aspects of Applied Sciences
	Intro to Programming Logic & Design (IT0 course)
	Introduction to Programming
Year 3	
	Data Applications & Interpretation
	Technical Communication
	Ethical & Legal Issues in IT
	Database Design
	Database Programming
	Web-Based Applications
	IST Electives – 4 courses
Year 4	
	Systems Analysis & Design
	IT Project Management
	Internship
	IST Electives – 6 courses

IST Tracks and Electives

Track: Network & Information Security	Track: Web Systems Development	Non-Track Electives
Information Assurance	Programming II	Android Application Development
Network Security	Server-Side Web Development	Application Development Environments
WAN Installation & Admin	Advanced Web Application Development	Assistive Technologies & Accessible Web Design
Enterprise Network Mgmt	Software Engineering & Mgmt	Cases in Information Systems Technology
Advanced Enterprise Net Mgmt		Database Administration
		Desktop Publishing Applications
		Health Information Technology
		Information Storage & Mgmt
		Intro to Video Game Design & Industry

ISSN: 2167-1435
v31 n3002

APPENDIX B
IT2008 Programming Fundamentals Units/Topics and Core Learning Outcomes in IT0

Topics	Core Learning Outcomes
Fundamental Programming Constructs*: 10 hours*	
• Basic syntax and semantics of a higher-level language • Variables, types, expressions, and assignment • Conditional and iterative control structures • Simple I/O • Functions and parameter passing • Structured decomposition • Recursion	1. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit. 2. Modify and expand short programs that use standard conditional and iterative control structures and functions. 3. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions. 4. Choose appropriate conditional and iteration constructs for a given programming task. 5. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces. 6. Describe the mechanics of parameter passing and the issues associated with scoping. 7. Describe the concept of recursion and give examples of its use.
Algorithms and Problem Solving*: 6 hours*	
• Problem solving strategies • The role of algorithms in the problem-solving process • Implementation strategies for algorithms • Debugging strategies • The concept and properties of algorithms	1. Discuss the importance of algorithms in the problem-solving process. 2. Identify the necessary properties of good algorithms. 3. Create algorithms for solving simple problems. 4. Use a programming language to implement, test, and debug algorithms for solving simple problems. 5. Apply effective debugging strategies.

ISSN: 2167-1435
v31 n3002

APPENDIX C
IT2008 Math and Statistics for IT Units/Topics and Core Learning Outcomes in IT0

Topics	Core Learning Outcomes
Basic Logic*: 10 hours*	
• Propositional logic • Logical connectives • Truth tables and validity • Predicate logic • Universal and existential quantification • Limitations of predicate logic	1. Apply formal methods of propositional and predicate logic. 2. Create a truth table to determine whether a given formula in predicate logic is valid. 3. Render a well-formed formula in predicate logic in English. 4. Explain the importance and limitations of predicate logic.
Functions, Relations and Sets: *6 hours*	
• Functions • Relations • Sets and set operations	1. Explain, with examples, the basic terminology of functions, relations, and sets. 2. Perform the standard operations associated with sets, functions, and relations. 3. Relate practical examples to the appropriate set, functions, or relation model, and interpret the associated operations and terminology in context.
Graphs and Trees*: 4 hours*	
• Trees • Undirected graphs • Directed graphs • Spanning trees • Traversal strategies	1. Illustrate, by example, the basic terminology of graph theory, and some of the properties and special cases of each type of graph. 2. Demonstrate different traversal methods for trees and graphs. 3. Model problems in IT using graphs and trees.

ISSN: 2167-1435
v31 n3002

APPENDIX D
IT0 Master Syllabus

MASTER SYLLABUS

COURSE NO., HOURS, AND TITLE: IST 207-3 Programming Logic and Design

COURSE DESCRIPTION:
This course provides students with the foundation for computer programming covering topics such as problem analysis, flowcharting, pseudocode, and algorithm development. Concepts such as documentations, structured design and modularity are emphasized. The course also introduces topics in discrete mathematics such as number systems, sets and logic, relations and functions, truth tables, trees, and graphs.

PREREQUISITES TO: IST 209

COURSE OBJECTIVES:

Upon successful completion of this course, the student will be able to:

1. Perform conversions and calculations with a variety of number systems
2. Apply formal methods of propositional and predicate logic.
3. Create a truth table to determine whether a given formula in predicate logic is valid.
4. Explain the basic terminology and perform of functions, relations, and sets.
5. Perform standard operations associated with functions, relations, and sets.
6. Illustrate, by example, the basic terminology of graph theory.
7. Demonstrate different traversal methods for trees and graphs.
8. Design structured problem solutions using tools such as flowcharts, pseudocode, and algorithms.
9. Demonstrate knowledge of input, output, variables, data types and validation.
10. Demonstrate knowledge of decision and repetition structures.
11. Demonstrate knowledge of programming functions and modularization.

TOPICAL OUTLINE:

Topics	**Percentages of Time**
I. Topics in Discrete Mathematics	
A. Number systems	4%
B. Propositional logic	4%
C. Truth tables and validity	4%
D. Predicate logic	4%
E. Functions	4%
F. Relations	4%
G. Sets and set operations	4%
H. Trees	4%
I. Graphs	4%

IST 207 Master Syllabus revised 02/18/2013

ISSN: 2167-1435
v31 n3002

II. Programming Logic and Design

A. Input, processing, and output	15%
B. Modules	10%
C. Decision structures	10%
D. Repetition structures	15%
E. Functions	10%
F. Input validation	4%

TEXTBOOKS:

Required:

Gaddis, T. (2013). Starting out with programming logic and design (3rd ed.) Upper Saddle River, NJ: Addison-Wesley.

IST 207 Master Syllabus revised 02/18/2013