School of Computer Science
Course Level Learning Outcomes

Sep 15, 2017 (Draft)

Notes:

- Course Level Learning Outcomes should be consistent no matter who is teaching the course and should be on the course outline.
- Course Level Learning Outcomes should not mention technology unless the course is specifically tied to that technology (e.g., CIS*1500 and C)
- There should be 4-7 learning outcomes per course.
    - More specific learning outcomes can be assigned to individual elements of the course to provide additional details. These should be linked to one of the course level learning outcomes.
- Learning outcomes should be described using active verbs. Lower level courses can have learning outcomes in the "knowledge" level. Higher level courses should have most learning outcomes at the Application or higher level of thinking and learning.

Knowledge	Comprehension	Application	Analysis	Synthesis	Evaluation
define	describe	apply	analyze	arrange	appraise
list	discuss	demonstrate	appraise	assemble	assess
name	explain	dramatize	calculate	collect	choose
recall	express	employ	categorize	compose	compare
record	depict	illustrate	criticize	construct	estimate

https://www.mnstate.edu/assess/poa/actionverbs.aspx

CIS*1250 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Discuss the properties of good software design including the nature and the role of associated documentation. 2) Create and specify the software design for a small-size software productizing a software requirement specification, an accepted program design methodology and appropriate design notation. 3) Explain the concept of a software life cycle and provide an example, illustrating its phases including the deliverables that are produced. 4) Use a common, non-formal method to model and specify (in the form of a requirements specification document) the requirements for a medium-size software system. 5) Demonstrate through involvement in a team project the central elements of team building and team management.

CIS*1500 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Demonstrate the role of algorithms in the problem solving process. 2) Demonstrate mastery of elementary C coding constructs by designing, implementing and testing C programs. 3) Utilize effective strategies for testing and program debugging. 4) Demonstrate elementary defensive programming techniques. 5) Describe how modular programming improves program design. 6) Identify improvements for command line programs with usability issues. 7) Use files and streams to input and output data.

CIS*1910 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Use Boolean algebra laws to derive other laws and to manipulate and complement Boolean expressions. 2) Relate Boolean algebra to logic and sets. 3) Apply formal methods of symbolic propositional and predicate logic, and informal but rigorous logical reasoning. 4) Give examples of the appropriate use of fundamental proof methods, including weak vs. strong induction. 5) Perform basic operations associated with sets, functions and relations. 6) Perform basic arithmetic operations in and conversions between binary, octal, hexadecimal, and decimal number systems. 7) Explain the use and importance of all the concepts and tools above in computer science and in modelling real-life situations.

CIS*2030 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Explain the digital representation of numeric and non-numeric data. 2) Identify the main abstracts that exist between programs and the actual hardware they run on, explain why these abstractions exist, and describe how they build upon each other. 3) Create assembly-language programs to implement some of the main abstractions and evaluate your program through testing and debugging. 4) Implement interfaces with external devices using common input-output strategies. 5) Describe the memory hierarchy and list trade-offs.

CIS*2430 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Identify the major characteristics of different programming paradigms (procedural, functional, logical, and object-oriented) and differentiate between procedural and object-oriented paradigms. 2) Design and implement classes for an object oriented program demonstrating correct use of encapsulation, constructors, method overloading, class invariants, accessors, mutators, instance variables and class variables. 3) Construct class hierarchies that maximize code reuse through inheritance while accommodating differences through method overriding. 4) Describe polymorphism and identify situations in which it is used in an OO program. 5) Use polymorphism, abstract methods/classes, and interfaces effectively to produce generic code. 6) Read and understand class diagrams written in UML (Unified Modeling Language). 7) Compare event-driven programming with control-driven programming. 8) Use layout managers, containers, components, listeners, anonymous classes and event handlers to create effect graphical user interfaces.

CIS*2500 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Create methodically tested and debugged programs that guard against input errors (user-proofing). 2) Correctly and effectively use dynamic and static memory allocation to solve computing tasks using C. 3) Create modular programs using functions and third party libraries that adhere to prescribed coding conventions and meet specification. 4) Construct well designed, reusable C language modules with appropriate data types (reading/writing binary and text files, and linked list). 5) Use programming tools effectively (git, make, gcc, pre-processor, gdb, valgrind, text editors).

CIS*2520 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Describe and implement common data structures for solving complex programming problems including algorithms for the creation, insertion, deletion, searching, and sorting of each data structure. 2) Analyse the space and time efficiency of algorithms including algorithms for the creation, insertion, deletion, searching, and sorting of data structures discussed. 3) Select and correctly use the appropriate abstract data type for programming problems.

CIS*2750 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Practice effective strategies for learning to use new software frameworks, methodologies, and programming languages. 2) Construct software components that adhere to provided specifications. 3) Integrate software components written in different programming languages to create a software system. 4) Design and implement software libraries. 5) Demonstrate systematic quality assurance and software testing techniques. 6) Employ a database or a file-based back end to implement data storage for an interactive program. 7) Identify and apply appropriate human-computer interaction techniques to the design of a graphical user interface.

CIS*2910 Course Level Learning Outcomes (4-7 high level outcomes per course)
1) Use basic counting techniques to enumerate/count basic combinatorial objects. 2) Determine the independence and probability of events using counting strategies. 3) Define basic graph theoretic terminology and graph representations. 4) Model a variety of computational problems using graphs.