Course Outcomes for each required course in the BSU CS major:

**Comp151 Computer Science I:**

At the end of the course students should be able to:

- understand the fundamental syntax & computer programs
- understand the fundamental control and loop (iteration) structures
- program simple algorithms, such as counting, summing, and finding maximum/minimum
- Implement, test, and debug simple recursive functions and procedures
- understand the basic data structures used in programming (such as arrays and array lists).
- Demonstrate knowledge of OOP concepts: instance variables, methods, objects and classes.
- argue effectively about the merits and possible unintended consequences of a computing implementation
- effectively write or present about the impact of computing on society. Students should extrapolate from historic lessons learned from unintended consequences of computing to the current computer solutions.

**Comp152 Computer Science II:**

After taking this course the student should be able to:

- Design, implement, and test the implementation of “is-a” relationships among objects using a class hierarchy and inheritance
- Design and implement programs that require a) multiple classes and structures b) hierarchies of classes that use inheritance and polymorphism
- Develop code that responds to exception conditions raised during execution
- Design, code, test, and debug simple event-driven programs that respond to user events
- Create classes with methods
- Discuss the consequences of software piracy on software developers and the role of relevant enforcement organizations
- Understand the uses and limitations of trademark, copyright and patents for intellectual property protection.
Comp 206 Introduction to Computer Organization:
After taking this course the student should be able to:
  • Design a simple circuit using fundamental building blocks.
  • Appreciate the effect of AND, OR, NOT and EOR operations on binary data
  • Understand how numbers, text, images, and sound can be represented in digital form and the limitations of such representations
  • Understand how errors due to rounding effects and their propagation affect the accuracy of chained calculations.
  • To understand the relationship between instruction set architecture, microarchitecture, and system architecture
  • and their roles in the development of the computer.
  • Be aware of the various classes of instruction: data movement, arithmetic, logical, and flow control.
  • Appreciate the difference between register-to-memory ISAs and load/store ISAs.
  • Appreciate how conditional operations are implemented at the machine level.
  • Understand the way in which subroutines are called and returns made.
  • Understand how, at the assembly language level, how parameters are passed to subroutines.

Comp330 Data Structures and Algorithms:
After taking this course the student should be able to:
  • Implement abstract data types in multiple ways recognizing the various strengths and weaknesses of those implementations.
  • Contrast different implementations through objective means, such as asymptotic analysis of the primitive functions.
  • Apply the concept of algorithmic efficiency to understand that different implementations may be “best”, depending on the requirements of the required task.

Comp340 Programming Languages:
After taking this course the student should be able to:
Course Objectives:
  • Explain the differences between different programming paradigms
  • Describe history and evolution of programming languages
  • Explain the abstractions (e.g., grammars) of programming languages
  • Demonstrate how control structures differ among languages
  • Describe the advantages/disadvantages of implicit versus explicit variable declarations
  • Explain the advantages/disadvantages of strong versus weak type checking
  • Describe how different languages implement encapsulation
. Understand the OO paradigm and how it contrasts to non-OO languages

. Explain how different languages (e.g., Java vs. C++) implement object orientation (single vs. multiple inheritance, polymorphism, etc.)

Comp350 Operating Systems:
After taking this course the student should be able to:
. Understand the fundamental concepts of operating systems
. Understand the concepts of process and thread provided in the modern operating systems with examples from Unix
. Understand process scheduling in a multi-programming environment and implement a process scheduling algorithm.
. Have practical experience with software tools available in modern operating systems such as semaphores and monitors for process synchronization
. Understand memory management techniques, including virtual memory in the modern operating systems
. Understand file system structure and implement a file system such as FAT

Comp430 Computer Networks:
After taking this course the student should be able to:
. Understand the basic concepts in computer networking
. Apply the knowledge of Mathematics in Computer Networking problems
. Understand the need for and structure of the OSI, TCP/IP network models
. Gain experience in the design and analysis of network protocols
. Develop solid client-server applications using TCP/IP
. Learn the principles of computer security and data protection
**Comp 435 Analysis of Algorithms**
After taking this course the student should be able to:

- recognize and use standard algorithms, adapt them to different problems, and analyze the efficiency of such adaptations.
- discuss the limitations of these algorithms, including any cases in which they fail as well as their worst-case behaviors.
- choose appropriate data structures to optimize algorithm performance.

**COMP 442 - Object-Oriented Software Engineering**
After taking this course the student should be able to:

- Use the wide range of possible software development models and their adaptive and hybrid application to specific cases.
- Demonstrate a practical understanding of the evolvability of user needs, and the challenges of communicating with software customers.
- Apply techniques for extracting user needs and creating semi-formal user needs analyses.
- Demonstrate practical experience with creating use cases, and considering both anticipated and unanticipated impacts.
- Formulate functional requirements based on current understanding of user needs.
- Understand principles and practices of good team management, collaboration, planning, and communication.
- Demonstrate experience with mapping functional requirements into alternative designs using available software development tools and environments, and assessing these alternatives.
- Demonstrate practical experience with implementing a complex, user-oriented software system.
- Apply iterative, agile software development practices.
- Understand the benefits of version control.
- Document their work, during design, and prior to delivery. And understand why this is necessary.
- Work effectively in teams
- Communicate effectively to other computer scientists.

**COMP 470 - Introduction to Artificial Intelligence**
After taking this course the student should be able to:

- Understand fundamental concepts and techniques in artificial intelligence, as well as a knowledge of the origins of the subject, its recent trends, and its ethical and societal implications.
- Demonstrate effective expository writing.
• Describe, design, and implement programming solutions to selected problems, involving intelligent agents, planning, knowledge representation, logic, probabilistic reasoning, learning, pattern recognition, natural language processing, and information extraction.
• Compare alternative procedural and non-procedural programming solutions.