The following updates to course titles, descriptions and prerequisites are proposed. For each course, the old information is shown with text to be changed shown in strikeout type. The proposed new information is shown with the additions underlined.

**CMSC 106**

(old) Introduction to C Programming; (4 credits) Grade Method: REG/P-F. Prerequisite: permission of department. Pre- or corequisite: MATH140. Only for CMPS, ENCP and students with major code: 2299F. Not open to students who have completed CMSC114 or higher. Design and analysis of programs in C. An introduction to computing using structured programming concepts. For further information contact the Undergraduate Education Office, Computer Science Department.

(new) Introduction to C Programming; (4 credits) Grade Method: REG/P-F. Permission of department required. Not open to students who have completed CMSC131 or higher. Design and analysis of programs in C. An introduction to computing using structured programming concepts.

**CMSC 131**

(old) Object-Oriented Programming I; (4 credits) Grade Method: REG/P-F. Pre- or corequisite: MATH140 and permission of department. Not open to students who have completed CMSC114. Introduction to programming and computer science. Emphasizes understanding and implementation of applications using object-oriented techniques. Develops skills such as program design and testing as well as implementation of programs using a graphical IDE. Programming done in Java. For CMSC majors only.

(new) Object-Oriented Programming I; (4 credits) Grade Method: REG. Pre- or corequisite: MATH140. Permission of department required. Introduction to programming and computer science. Emphasizes understanding and implementation of applications using object-oriented techniques. Develops skills such as program design and testing as well as implementation of programs using a graphical IDE. Programming done in Java.

**CMSC 132**

(old) Object-Oriented Programming II; (4 credits) Grade Method: REG/P-F/AUD. Prerequisite: CMSC131 with a grade of C or better; or a score of 5 on the A Java AP exam; or a score of 4 or 5 on the AB Java AP exam; or permission of the department based on satisfactory performance on the department placement exam and permission of department. Corequisite: MATH141. Introduction to use of computers to solve problems using software engineering principles. Design, build, test, and debug medium-size software systems and learn to use relevant tools. Use object-oriented methods to create effective and efficient problem solutions. Use and implement application programming interfaces (APIs). Programming done in Java.

(new) Object-Oriented Programming II; (4 credits) Grade Method: REG. Prerequisite: CMSC131 with a grade of C or better; or a score of 5 on the A Java AP exam; or satisfactory performance on the department placement exam and MATH140 with a grade of C or better. Permission of department required. Introduction to use of computers to solve problems using software engineering principles. Design, build, test, and debug medium-size software systems and learn to use relevant tools. Use object-oriented methods to create effective and efficient problem solutions. Use and implement application programming interfaces (APIs). Programming done in Java.

**CMSC 216**

Create new course (see attached materials)

**CMSC 330**
(old) Organization of Programming Languages; (3 credits) Grade Method: REG. Prerequisite: CMSC212 and CMSC250: each with a grade of C (2.0) or better; and permission of department. The semantics of programming languages and their run-time organization. Several different models of languages are discussed, including procedural (e.g., C, Pascal), functional (e.g., ML, Lisp), rule-based (e.g., Prolog), and object-oriented (e.g., C++, Smalltalk). Run-time structures, including dynamic versus static scope rules, storage for strings, arrays, records, and object inheritance are explored.

(new) Organization of Programming Languages; (3 credits) Grade Method: REG. Prerequisite: CMSC216 and CMSC250: each with a grade of C (2.0) or better; and permission of department. Introduction to programming languages. Topics include formal syntax, functional programming, type systems, scoping and binding of variables, and concurrency.

CMSC 351
(old) CMSC351 (PermReq) Algorithms; (3 credits) Grade Method: REG. Prerequisite: CMSC212 and CMSC250: each with a grade of C (2.0) or better; and permission of department. A systematic study of the complexity of some elementary algorithms related to sorting, graphs and trees, and combinatorics. Algorithms are analyzed using mathematical techniques to solve recurrences and summations.

(new) CMSC351 (PermReq) Algorithms; (3 credits) Grade Method: REG. Prerequisite: CMSC216 and CMSC250: each with a grade of C (2.0) or better; and permission of department. A systematic study of the complexity of some elementary algorithms related to sorting, graphs and trees, and combinatorics. Algorithms are analyzed using mathematical techniques to solve recurrences and summations.

CMSC 411
(old) CMSC411 Computer Systems Architecture; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC311 or ENEE350 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Input/output processors and techniques. Intra-system communication, buses, caches. Addressing and memory hierarchies. Microprogramming, parallelism, and pipelining.

(new) CMSC411 Computer Systems Architecture; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC216 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Pipelining, instruction-level parallelism, multiprocessors. Memory hierarchy, caches, storage systems. Interprocessor communication, buses and networks.

CMSC 412
(old) Operating Systems; (4 credits) Grade Method: REG. CORE Capstone (CS) Course. Prerequisites: A grade of C or better in CMSC311 or ENEE350 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. An introduction to batch systems, spooling systems, and third-generation multiprogramming systems. Description of the parts of an operating system in terms of function, structure, and implementation. Basic resource allocation policies.

(new) Operating Systems; (4 credits) Grade Method: REG. CORE Capstone (CS) Course. Prerequisites: A grade of C or better in CMSC216 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Design and implementation of operating system functions including processes, concurrency, synchronization, memory management, resource allocation, and file systems.

CMSC414
(old) CMSC414 Computer and Network Security; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC311 or ENEE350 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. An introduction to the topic of security in
the context of computer systems and networks. Identify, analyze, and solve network-related security problems in computer systems. Fundamentals of number theory, authentication, and encryption technologies, as well as the practical problems that have to be solved in order to make those technologies workable in a networked environment, particularly in the wide-area Internet environment.

(new) CMSC414 Computer and Network Security; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC216 and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. An introduction to the topic of security in the context of computer systems and networks. Identify, analyze, and solve network-related security problems in computer systems. Fundamentals of number theory, authentication, and encryption technologies, as well as the practical problems that have to be solved in order to make those technologies workable in a networked environment, particularly in the wide-area Internet environment.

CMSC417
(old) CMSC417 Computer Networks; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC351, a grade of C or better in (CMSC311 or ENEE350), and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Computer networks and architectures. The OSI model including discussion and examples of various network layers. A general introduction to existing network protocols. Communication protocol specification, analysis, and testing.

(new) CMSC417 Computer Networks; (3 credits) Grade Method: REG. Prerequisite: A grade of C or better in CMSC351, and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Computer networks and architectures. The OSI model including discussion and examples of various network layers. A general introduction to existing network protocols. Communication protocol specification, analysis, and testing.

CMSC 423
(old) CMSC423 Bioinformatic Algorithms, Databases, and Tools; (3 credits) Grade Method: REG/P-F/AUD. Prerequisite: CMSC351 or permission of department. A practical introduction to the main topics in algorithms, databases, and tools used in bioinformatics. Includes public databases such as Genbank and PDG, software tools such as BLAST, and their underlying algorithms. Use of Perl scripting language to perform a number of useful tasks in analyzing sequence data and managing bioinformatic databases.

(new) CMSC423 Bioinformatics; (3 credits) Grade Method: REG/AUD. Prerequisite: CMSC351 or permission of department. An introduction to the main algorithms, databases, and tools used in bioinformatics. Topics may include assembly and analysis of genome sequences, reconstructing evolutionary histories, predicting protein structure, and clustering of biological data. Use of scripting languages to perform analysis tasks on biological data. No prior knowledge of biology is assumed.

CMSC 424
(old) CMSC424 Database Design; (3 credits) Grade Method: REG. CORE Capstone (CS) Course. Prerequisite: CMSC420 with a grade of C or better, and permission of department; or CMSC graduate student. Motivation for the database approach as a mechanism for modeling the real world. Review of the three popular data models: relational, network, and hierarchical. Comparison of permissible structures, integrity constraints, storage strategies, and query facilities. Theory of database design logic.

(new) CMSC424 Database Design; (3 credits) Grade Method: REG. CORE Capstone (CS) Course. Prerequisite: A grade of C or better in CMSC351, and a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. This course introduces students to database systems and motivates the database approach as a mechanism for modeling the real world.
The course covers in depth the relational model, logical database design, query languages, and other database concepts including query optimization, concurrency control, transaction management, and log-based crash recovery. Distributed and Web database architectures are also discussed.

CMSC 430
(old) CMSC430 Theory of Language Translation; (3 credits) Grade Method: REG. Prerequisites: a grade of C or better in CMSC330; and permission of department; or CMSC graduate student. Formal translation of programming languages, program syntax and semantics. Finite state recognizers and regular grammars. Context-free parsing techniques such as recursive descent, precedence, LL(k) and LR(k). Code generation, improvement, syntax-directed translation schema.

(new) CMSC430 Introduction to Compilers; (3 credits) Grade Method: REG. Prerequisites: a grade of C or better in CMSC330; and permission of department; Topics include lexical analysis, parsing, intermediate representations, program analysis, optimization, and code generation.

CMSC 630
(old) CMSC630 Theory of Programming Languages; (3 credits) Grade Method: REG/AUD. Prerequisite: CMSC 430. Contemporary topics in the theory of programming languages. Formal specification and program correctness. Axiomatic proof systems (both Floyd-Hoare and Dijkstra’s predicate transformers), Mills’ functional correctness approach, abstract data types (both abstract model and algebraic specifications), and Scott-style denotational semantics based on least fixed points.