DUM DUM MOTIJHEEL COLLEGE

DEPARTMENT OF COMPUTER SCIENCE

B.Sc. Computer Science (Hons) CBCS Syllabus

Issued by the West Bengal StateUniversity

With effect from 2018-19

Programme Specific Outcomes

- After Successful completion of undergraduate program in Computer Sciecne students will gain a strong understanding of fundamental concepts in computer science, including algorithms, data structures, computer organization, and software engineering principles.
- Develop proficiency in multiple programming languages and the ability to design, implement, and debug complex software systems.
- Acquire strong problem-solving skills and the ability to apply computational thinking to solve real-world problems.
- Gain a solid understanding of theoretical aspects of computer science, including formal languages, automata theory, and complexity theory.
- Learn the principles of database design, implementation, and management, including relational database systems.
- Understand the fundamentals of computer networks and cyber security, including protocols, network design, and security measures.
- Gain knowledge about the design and functioning of operating systems, including process management, memory management, and file systems.
- Explore concepts and techniques in artificial intelligence, machine learning, and data science.
- Understand the entire software development lifecycle, from requirements analysis and design to testing, deployment, and maintenance.
- Foster the ability to engage in research, stay updated with the latest advancements in technology, and contribute to innovation in the field.

DUM DUM MOTIJHEEL COLLEGE
Course Outcome or Learning Outcome
Three year B.A. /B.Sc. degree course
Under CBCS semester system
HONOURS COURSE IN COMPUTER SCIENCE
With effect from the session: 2018 – 2019

Course Name: Core Course-1

Course Code: CMSACOR01T and CMSACOR01P
Topic Name: Programming Fundamental using C/C++

Course Outcome: After completion of this course student will understand and apply fundamental programming constructs such as variables, data types, operators, loops, and conditional statements in both C and C++. earn to define and use functions for code modularity, reusability, and maintainability. Understand the concept of parameter passing and return values. Introduce basic data structures like arrays, linked lists, stacks, and queues. Learn how to manipulate and organize data efficiently. Understand memory allocation and deallocation in C/C++ and gain insights into pointers and dynamic memory allocation. Explore file I/O operations in C/C++ to read from and write to files. Understand how to manipulate data stored in files. students will learn about OOP principles, including classes, objects, inheritance, polymorphism, encapsulation, and abstraction. Develop problem-solving skills through the implementation of algorithms and logical thinking using C/C++ programming constructs. Acquire skills in debugging code and handling errors effectively. Learn to use debugging tools and techniques. Understand and apply coding standards and best practices for writing clean, readable, and maintainable code. Gain an understanding of basic algorithms and their efficiency. Practice algorithmic thinking to solve computational problems. Apply learned concepts through hands-on projects. This may involve writing small to medium-sized programs to solve real- world problems, reinforcing the practical application of programming skills.

Course Name: Core Course-2

Course Code: CMSACOR02T and CMSACOR02P
Topic Name: Computer System Architecture

Course Outcome: From this course students will gain a deep understanding of the organization and components of a computer system, including the CPU, memory, input/output devices, and the interconnection structure. Learn about instruction sets, addressing modes, and the design principles of the instruction set architecture. Understand how instructions are executed by the CPU. Explore the design and microarchitecture of processors, including pipelining, instruction-level parallelism, and techniques for improving CPU performance. Understand the memory hierarchy, including cache memory, main memory, and secondary storage. Learn about memory management techniques and their impact on system performance. Study the principles of input/output systems, including I/O interfaces, interrupt handling, and data transfer mechanisms between the CPU and peripherals. Explore bus systems and interconnection networks that facilitate communication between different components of a computer system. Understand the principles of parallel and distributed computing, including multi-core processors, parallel architectures, and the challenges of distributed systems. Gain hands-on experience with assembly language programming to reinforce the understanding of computer architecture concepts. Learn techniques for performance evaluation and optimization of computer systems, including benchmarking, profiling, and tuning. Stay updated with current trends and advancements in computer architecture, including emerging technologies and architectures.

Course Name: **Core Course-3**

Course Code: **CMSACOR03T and CMSACOR03P**
Topic Name: **Programming in Java**

Course Outcome: Students will understand the basic syntax, data types, and control flow structures in Java, including variables, loops, and conditional statements. Gain a solid foundation in OOP principles, including classes, objects, inheritance, polymorphism, encapsulation, and abstraction. Familiarize yourself with the Java Standard Library (Java API) and learn how to use pre-built classes and methods for common programming tasks. Understand the concept of exceptions in Java and learn how to handle errors and exceptions in a program. Explore basic GUI programming using libraries such as Swing or JavaFX. Learn to create interactive and user-friendly applications. Gain proficiency in reading from and writing to files, understanding file handling operations in Java. Learn about the Java Collections Framework, including data structures such as lists, sets, and maps. Understand how to use these structures for efficient data manipulation. Understand the basics of multithreading in Java, including creating and managing multiple threads to achieve concurrent execution. Explore networking concepts and how to implement networked applications in Java, including socket programming.

Course Name: **Core Course-4**

Course Code: **CMSACOR04T and CMSACOR04P**
Topic Name: **Discrete Structure**

Course Outcome: Students will understand the basics of set theory, including set operations, subsets, and set relations. Apply set theory concepts to solve problems and model real-world situations. Learn the principles of propositional logic, including logical operators, truth tables, and logical equivalences. Develop skills in constructing and analyzing logical expressions. Extend logical reasoning to predicate logic, including quantifiers and proofs. Understand how to express statements about elements in a set using predicate logic. Acquire proficiency in various proof techniques, such as direct proofs, proof by contrapositive, proof by contradiction, and mathematical induction. Explore fundamental concepts in number theory, including divisibility, prime numbers, greatest common divisors, and modular arithmetic. Understand counting principles, permutations, combinations, and the binomial theorem. Apply combinatorial concepts to solve problems related to arrangements and selections. Study basic concepts in graph theory, including vertices, edges, paths, cycles, and connectivity. Analyze and solve problems related to graph representations. Explore the concepts of relations and functions, including equivalence relations, partial orders, and injections/surjections. Understand how to represent and analyze relationships between sets. Introduce the concept of finite state machines and understand their application in modeling and solving problems related to computation. Learn the basics of formal languages, regular languages, and finite automata. Understand how automata theory is connected to the study of programming languages and computation. Introduce basic concepts of algorithmic complexity and analyze the efficiency of algorithms in terms of time and space complexity. Relate discrete structures to their applications in computer science, such as algorithm design, database management, and network design.

Course Name: **Core Course-5**

Course Code: **CMSACOR05T and CMSACOR05P**
Topic Name: **Data Structure**

Course Outcome: Students will gain a solid understanding of fundamental data structures such as arrays, linked lists, stacks, and queues. Learn to analyze the time and space complexity of algorithms, with a focus on understanding the efficiency of data structure operations. Study hierarchical data structures, including binary trees, binary search trees, and AVL trees. Understand operations on trees and their applications. Explore graph data structures and algorithms, including graph traversal, shortest path algorithms, and graph connectivity. Understand the concept of hashing and hash tables. Learn how to implement and use hash functions to achieve efficient data retrieval. Study heap data structures and their applications, including priority queues and heap-based sorting algorithms. Explore more advanced data structures, such as tries and B-trees, and understand their applications in efficient data storage and retrieval.

Course Name: **Core Course-6**

Course Code: **CMSACOR06T and CMSACOR06P**
Topic Name: **Operating System**

Course Outcome: Define what an operating system is and explain its role in managing computer hardware and software resources. Describe the evolution of operating systems and their key components. Understand the concept of a process and the role of process management in an OS. Learn about process scheduling algorithms and how they impact system performance. Explain memory hierarchy and the role of the operating system in managing different types of memory. Understand virtual memory concepts, paging, and segmentation. Describe file system organization and structure. Learn about file operations, directory structures, and file access control. Understand I/O devices and their interaction with the operating system. Learn about I/O scheduling and how it impacts system performance. Explore issues related to concurrent processes and the need for synchronization. Study synchronization mechanisms, such as locks and semaphores. Identify and analyze deadlock situations in a system. Learn strategies for deadlock prevention, avoidance, and recovery. Understand the importance of security in operating systems. Learn about access control, authentication, and other security mechanisms. Analyze real-world operating systems as case studies. Gain insights into the design principles and trade-offs in building operating systems. Evaluate the performance of an operating system and identify bottlenecks. Explore contemporary topics and emerging trends in operating systems, such as virtualization, cloud computing, and containerization.

Course Name: **Core Course-7**

Course Code: **CMSACOR07T and CMSACOR07P**
Topic Name: **Computer Networks**

Course Outcome: Define computer networks and understand their importance in modern computing. Explore the historical development and evolution of computer networks. Study different network topologies and their advantages and disadvantages. Understand various networking protocols and their roles in communication. Explore the functionalities of the physical and data link layers of the OSI model. Learn about error detection and correction mechanisms. Understand the role of the network layer in routing and forwarding data. Study common routing algorithms and protocols. Explore the functions of the transport layer in end-to-end communication. Learn about flow control, error recovery, and congestion control mechanisms. Examine the application layer protocols and their role in supporting network applications. Study common application layer protocols, such as HTTP, FTP, and DNS. Understand the challenges and protocols associated with wireless and mobile communication. Explore technologies like Wi-Fi, Bluetooth, and cellular networks. Learn about common security threats in computer networks. Explore security mechanisms, such as firewalls, encryption, and intrusion detection systems. Understand the principles of network management and monitoring. Analyze real-world case studies of successful network implementations. Apply theoretical knowledge to practical scenarios. Gain practical experience through hands-on labs and projects. Configure and troubleshoot network setups to reinforce theoretical concepts.

Course Name: **Core Course-8**

Course Code: **CMSACOR08T and CMSACOR08P**
Topic Name: **Design & Analysis of Algorithm**
Course Outcome: Understand the fundamental concepts of algorithms, including correctness, efficiency, and optimality. Learn how to express algorithms using pseudocode or a programming language. Explore various algorithm design paradigms, such as divide and conquer, dynamic programming, and greedy algorithms. Understand when to apply each design technique to solve specific types of problems. Learn how to analyze the time complexity of algorithms using Big-O notation. Understand the importance of worst-case, average-case, and best-case analysis. Understand the concept of recursion and its application in algorithm design. Learn how to use backtracking to solve problems with multiple decision points. Study and implement various sorting algorithms, such as quicksort, mergesort, and heapsort. Explore searching algorithms, including binary search. Understand and implement graph traversal algorithms (e.g., depth-first search, breadth-first search). Learn about graph algorithms for shortest paths and minimum spanning trees. Explore dynamic programming as a technique for solving optimization problems. Implement dynamic programming solutions for various problems. Learn the principles of greedy algorithms and when to use them. Implement greedy algorithms for solving optimization problems. Understand the concept of NP- completeness and the implications for algorithmic problem-solving. Learn about reductions and the importance of solving hard problems efficiently. Explore algorithms that use randomness for solving problems. Understand the concept of probabilistic analysis. Learn about approximation algorithms for NP-hard optimization problems. Understand the trade-off between optimality and efficiency. Apply algorithmic design and analysis techniques to solve real-world problems. Practice translating problems into algorithmic solutions. Develop the ability to articulate algorithmic solutions clearly in both written and oral formats. Present and discuss algorithmic solutions in a structured manner.

Course Name: **Core Course-9**

Course Code: **CMSACOR09T and CMSACOR09P**
Topic Name: **Software Engineering**

Course Outcome: Understand the fundamental concepts and importance of software engineering in the development life cycle. Explore the key activities involved in software engineering processes. Learn about various models of SDLC, such as Waterfall, Agile, and Iterative models. Understand the strengths and weaknesses of different development methodologies. Study the process of gathering, analyzing, and documenting software requirements. Learn how to manage and prioritize requirements throughout the project. Understand the principles of system design, including architectural design and detailed design. Learn how to create design documentation and make design decisions. Explore best practices for coding, including code readability, maintainability, and documentation. Understand how to implement designs into executable code. Learn various testing techniques and strategies to ensure software quality. Understand the importance of test planning and test documentation. Study the challenges and techniques involved in software maintenance. Understand version control and configuration management. Learn project management principles specific to software engineering. Understand how to estimate project effort, schedule tasks, and manage resources. Explore metrics for measuring software quality, productivity, and performance. Understand how to use metrics for project tracking and improvement. Identify and analyze potential risks in software projects. Learn risk mitigation and contingency planning strategies. Understand collaborative development practices, including version control systems (e.g., Git). Learn about tools for collaboration, issue tracking, and continuous integration. Explore ethical considerations in software engineering. Understand the responsibilities of software engineers in a professional context. Emphasize the importance of clear and effective communication in software development. Learn how to create and maintain project documentation.

Course Name: **Core Course-10**

Course Code: **CMSACOR10T and CMSACOR10P**
Topic Name: **DBMS**

Course Outcome: The Database Management System (DBMS) course aims to equip students with the knowledge and skills necessary to understand, design, and manage databases effectively. Students typically learn fundamental concepts of database systems, including data modeling, normalization, and query languages. They gain proficiency in designing relational databases, creating and manipulating database schemas, and implementing complex queries using SQL. The course also covers topics such as indexing, transaction management, and database security. Additionally, students are introduced to various types of database models and gain insights into emerging trends in database technologies. By the end of the course, students are expected to be capable of designing and implementing robust database solutions, ensuring data integrity, and optimizing database performance in real-world applications. The understanding of database management principles acquired in this course is essential for students pursuing careers in software development, data analysis, and other fields where efficient and organized data storage and retrieval are critical.

Course Name: **Core Course-11**

Course Code: **CMSACOR11T and CMSACOR11P**
Topic Name: **Internet Technology**

Course Outcome: The Internet Technology course with a focus on JSP (JavaServer Pages) and JavaScript is designed to provide students with a comprehensive understanding of web development technologies. Throughout the course, students typically learn the basics of HTML, CSS, and client-side scripting with JavaScript to create interactive and dynamic user interfaces. The integration of JSP into the curriculum allows students to explore server-side programming, dynamic content generation, and the development of web applications using Java. They gain proficiency in using JSP to connect the front-end with back-end services and databases. Additionally, students learn to enhance their skills in creating responsive and user-friendly web applications. By the end of the course, students are expected to have a strong grasp of both client-side and server-side web development, enabling them to design and implement robust, interactive, and scalable web applications using a combination of JSP and JavaScript technologies. This skill set is particularly valuable for individuals pursuing careers in web development and application programming.

Course Name: **Core Course-12**

Course Code: **CMSACOR12T and CMSACOR12P**
Topic Name: **Theory of Computation**

Course Outcome: The Theory of Computation course is designed to deepen students' understanding of the theoretical foundations of computation and the limits of what can be computed. Throughout the course, students typically explore formal languages, automata theory, and computational complexity. They learn about different models of computation, including finite automata, pushdown automata, and Turing machines, and study their relationships with formal language classes such as regular and context-free languages. The course delves into the concept of algorithmic decidability, addressing questions related to the solvability of problems and the existence of algorithms. Students also engage with computational complexity theory, investigating the efficiency and inherent difficulty of solving problems within various computational models. By the end of the course, students are expected to develop a profound understanding of the theoretical underpinnings of computation, enabling them to analyze the computational complexity of algorithms and reason about the boundaries of computation in both practical and abstract contexts. This knowledge is crucial for individuals pursuing careers in theoretical computer science, algorithm design, and advanced software development.

Course Name: **Core Course-13**

Course Code: **CMSACOR13T and CMSACOR13P**
Topic Name: **Artificial Intelligence**

Course Outcome: The Artificial Intelligence (AI) course aims to impart students with a deep understanding of the principles, methodologies, and applications of artificial intelligence. Throughout the course, students typically explore key topics such as machine learning, natural language processing, computer vision, and expert systems. They learn the foundations of AI algorithms, including supervised and unsupervised learning, and gain practical experience in designing and implementing AI models. The course often includes hands-on projects that involve solving real-world problems using AI techniques. Additionally, students may delve into ethical considerations and societal impacts of AI applications. By the end of the course, students are expected to be proficient in leveraging AI technologies, capable of developing intelligent systems, and understanding the potential and challenges associated with artificial intelligence. This knowledge prepares students for diverse career paths in AI research, machine learning engineering, data science, and related fields where expertise in artificial intelligence is increasingly in demand.

Course Name: **Core Course-14**

Course Code: **CMSACOR14T and CMSACOR14P**
Topic Name: **Computer Graphics**

Course Outcome: The Computer Graphics course is designed to equip students with a comprehensive understanding of the principles, techniques, and applications in the field of computer graphics. Throughout the course, students typically delve into fundamental concepts such as 2D and 3D transformations, rendering, shading, and illumination models. They gain hands-on experience in programming graphics applications and working with graphics libraries and tools. The course often covers topics like raster graphics, vector graphics, and image processing. Students also explore computer animation, virtual reality, and graphical user interface (GUI) design. By the end of the course, students are expected to be proficient in creating visually appealing graphics, understanding the underlying mathematical concepts, and applying their knowledge to solve problems in areas such as gaming, simulation, design, and multimedia. This skill set is valuable for individuals pursuing careers in graphics programming, user interface design, and other fields where a strong foundation in computer graphics is essential.

Course Name: **DISCIPLINE SPECIFIC ELECTIVE**

Course Code: **CMSADSE01T and CMSADSE01P**
Topic Name: **Microprocessor**

Course Outcome: The Microprocessor course with a focus on 8085 and 8086 architectures is designed to equip students with in-depth knowledge of these popular microprocessors and their applications. Throughout the course, students typically explore the internal architecture, instruction set, and programming techniques specific to the 8085 and 8086 microprocessors. They learn assembly language programming, memory interfacing, and input/output operations for these processors. Hands-on experience with simulators or actual hardware is often incorporated, allowing students to implement and test programs on 8085 and 8086 microprocessor-based systems. The course may cover advanced topics such as interfacing with peripherals, interrupt handling, and system-level design using these microprocessors. By the end of the course, students are expected to be proficient in programming and interfacing with both 8085 and 8086 microprocessors, enabling them to design and implement embedded systems and understand the intricacies of microprocessor-based architectures. This skill set is valuable for individuals pursuing careers in embedded systems development, hardware design, and related fields where knowledge of specific microprocessor architectures is crucial.

Course Name: **DISCIPLINE SPECIFIC ELECTIVE**

Course Code: **CMSADSE02T and CMSADSE02P**
Topic Name: **Data Mining**

Course Outcome: The Data Mining course with a focus on WEKA (Waikato Environment for Knowledge Analysis) is designed to provide students with a comprehensive understanding of data mining techniques and tools, with an emphasis on practical applications using WEKA. Throughout the course, students typically learn the fundamental concepts of data mining, including data preprocessing, classification, clustering, association rule mining, and feature selection. The integration of WEKA into the curriculum allows students to gain hands-on experience in applying these techniques to real-world datasets. They learn to navigate the WEKA environment, perform data analysis, and interpret results. Additionally, students may explore advanced topics such as ensemble methods and evaluation metrics for model performance. By the end of the course, students are expected to be proficient in using WEKA for various data mining tasks, enabling them to extract valuable insights from large datasets and make informed decisions in fields such as business intelligence, healthcare, and research. This skill set is particularly valuable for individuals pursuing careers in data analysis, machine learning, and business analytics.

Course Name: **DISCIPLINE SPECIFIC ELECTIVE**

Course Code: **CMSADSE03T and CMSADSE03P**
Topic Name: **Cloud Computing**

Course Outcome: The Cloud Computing course is designed to equip students with a comprehensive understanding of the principles, technologies, and applications of cloud computing. Throughout the course, students typically delve into the fundamental concepts of cloud computing, including virtualization, service models (Infrastructure as a Service, Platform as a Service, Software as a Service), and deployment models (public, private, hybrid, and community clouds). Students learn about popular cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform, gaining hands-on experience in deploying and managing applications in the cloud. The course often covers topics like cloud security, scalability, and cost management. By the end of the course, students are expected to be proficient in designing, deploying, and managing cloud-based solutions, making them well-prepared for roles in cloud architecture, system administration, and application development in the rapidly evolving field of cloud computing. This skill set is crucial for individuals seeking careers where the ability to leverage cloud technologies is essential for scalable and efficient computing solutions.

Course Name: **DISCIPLINE SPECIFIC ELECTIVE**

Course Code: **CMSADSE05T and CMSADSE05P**
Topic Name: **Digital Image Processing**

Course Outcome: The Digital Image Processing course is designed to equip students with a thorough understanding of the principles, techniques, and applications of processing digital images. Throughout the course, students typically explore fundamental concepts such as image enhancement, restoration, segmentation, and compression. They learn about various image processing algorithms and tools used for manipulating and analyzing digital images. The course often covers topics like filtering, feature extraction, and pattern recognition within the context of image processing. Students gain hands-on experience in implementing these techniques using software tools and programming languages. By the end of the course, students are expected to be proficient in applying image processing methods to solve practical problems, enhance image quality, and extract valuable information from digital images. This skill set is valuable for individuals pursuing careers in computer vision, medical imaging, multimedia, and other fields where the manipulation and analysis of digital images play a crucial role.

Course Name: **DISCIPLINE SPECIFIC ELECTIVE**

Course Code: **CMSADSE06P**
Topic Name: **Project**

Course Outcome: This option to be offered only in 6th Semester. The students will be allowed to work on any project based on the concepts studied in core / elective or skill based elective courses. Typically a project-focused course in Software Engineering or Artificial Intelligence is availed and designed to provide students with hands-on experience in applying the theoretical concepts learned throughout their coursework. In a Software Engineering project, students typically work collaboratively to design, develop, and implement a software solution, emphasizing the entire software development life cycle. This includes requirements analysis, system design, coding, testing, and deployment, while also considering project management aspects. On the other hand, in an AI project, students often engage in creating intelligent systems, developing machine learning models, and solving real-world problems using AI techniques. They may explore diverse AI applications such as natural language processing, computer vision, or recommendation systems. By the end of the project, students are expected to showcase not only technical proficiency but also effective teamwork, project management skills, and the ability to apply their knowledge to solve complex problems. These project outcomes are valuable for preparing students to enter the workforce with practical experience, demonstrating their ability to bring theoretical concepts into practical fruition in the fields of Software Engineering or Artificial Intelligence.

Course Name: **SKILL ENHANCEMENT COURSE**

Course Code: **CMSSSEC01M**
Topic Name: **Programming in Python**

Course Outcome: Understand the basic syntax and structure of the Python programming language. Learn how to write and execute simple Python programs. Explore various data types in Python, such as integers, floats, strings, and booleans. Understand how to declare and manipulate variables. Learn about control flow structures, including if statements, loops, and conditional statements. Understand how to use these statements to control the flow of a program. Define and use functions to modularize code. Explore function parameters, return values, and the concept of scope. Introduction to fundamental data structures in Python, including lists, tuples, sets, and dictionaries. Learn how to perform operations on these data structures. Understand how to read from and write to files using Python. Explore file input/output operations. Learn the basics of exception handling to deal with errors gracefully. Understand try, except, and finally blocks. Introduction to OOP principles, including classes and objects. Learn how to create and use classes in Python. Understand how to use Python modules and libraries to extend functionality. Explore commonly used libraries, such as NumPy for numerical computing and matplotlib for data visualization. Introduction to web development using frameworks like Flask or Django (if covered in the course). Understand the basics of creating web applications with Python. Develop problem-solving skills using Python. Understand algorithmic thinking and basic algorithm design. Introduction to version control systems, particularly Git. Learn basic Git commands for collaborative coding and project management. Emphasize good coding practices, code readability, and style conventions. Understand the importance of documentation.

Course Name: **SKILL ENHANCEMENT COURSE**

Course Code: **CMSSSEC02M**
Topic Name: **R Programming**

Course Outcome: The course in R Programming aims to impart students with a comprehensive understanding of the R programming language and its applications in statistical analysis, data visualization, and data manipulation. Throughout the course, students typically learn the fundamentals of R, including syntax, data structures, and control flow. Emphasis is placed on statistical analysis using R, enabling students to conduct hypothesis testing, regression analysis, and exploratory data analysis. The course often covers data visualization techniques using packages like ggplot2, enhancing students' abilities to communicate insights effectively. Students may also gain proficiency in data manipulation tasks with tools like the dplyr package. Additionally, the course may introduce concepts related to reproducibility and version control, fostering good coding practices. By the end of the course, students are expected to be well-equipped to use R as a powerful tool for statistical computing and data analysis in various domains, including academia, research, and industry.

DUM DUM MOTIJHEEL COLLEGE
Course Outcome or Learning Outcome
Three year B.A. /B.Sc. degree course
Under CBCS semester system
GENERAL COURSE IN COMPUTER SCIENCE

With effect from the session: 2018 – 2019

Course Code: **CMSGCOR01T and CMSGCOR01P**
Topic Name: **Problem Solving with Computer**

Course Outcome: From this course students will gain a deep understanding of the organization and components of a computer system, including the CPU, memory, input/output devices, and the interconnection structure. Learn about instruction sets, addressing modes, and the design principles of the instruction set architecture. Understand how instructions are executed by the CPU. Explore the design and microarchitecture of processors, including pipelining, instruction-level parallelism, and techniques for improving CPU performance. Understand the memory hierarchy, including cache memory, main memory, and secondary storage. Learn about memory management techniques and their impact on system performance. Study the principles of input/output systems, including I/O interfaces, interrupt handling, and data transfer mechanisms between the CPU and peripherals. Explore bus systems and interconnection networks that facilitate communication between different components of a computer system. Understand the principles of parallel and distributed computing, including multi-core processors, parallel architectures, and the challenges of distributed systems. Gain hands-on experience with assembly language programming to reinforce the understanding of computer architecture concepts. Learn techniques for performance evaluation and optimization of computer systems, including benchmarking, profiling, and tuning. Stay updated with current trends and advancements in computer architecture, including emerging technologies and architectures.

Course Code: **CMSGCOR02T and CMSGCOR02P**
Topic Name: **DBMS**

Course Outcome: The Database Management System (DBMS) course aims to equip students with the knowledge and skills necessary to understand, design, and manage databases effectively. Students typically learn fundamental concepts of database systems, including data modeling, normalization, and query languages. They gain proficiency in designing relational databases, creating and manipulating database schemas, and implementing complex queries using SQL. The course also covers topics such as indexing, transaction management, and database security. Additionally, students are introduced to various types of database models and gain insights into emerging trends in database technologies. By the end of the course, students are expected to be capable of designing and implementing robust database solutions, ensuring data integrity, and optimizing database performance in real-world applications. The understanding of database management principles acquired in this course is essential for students pursuing careers in software development, data analysis, and other fields where efficient and organized data storage and retrieval are critical.

Course Code: **CMSGCOR03T and CMSGCOR03P**
Topic Name: **OS**

Course Outcome: The course integrating Operating System Theory with practical Linux applications is designed to offer students a holistic understanding of both theoretical concepts and real-world implementation. Throughout the course, students delve into the fundamental principles of operating systems, covering topics such as process management, memory management, file systems, and system calls. The practical aspect involves hands-on experience with Linux, enabling students to apply theoretical knowledge to a Unix-like operating system. Students gain proficiency in using Linux commands, writing shell scripts, and performing system administration tasks. The course often includes projects where students configure, manage, and troubleshoot Linux systems, reinforcing theoretical concepts in a practical context. By the end of the course, students are expected to possess a solid theoretical foundation in operating systems while also being adept at navigating and utilizing Linux environments for system-related tasks. This dual focus prepares students for careers in system administration, software development, and related fields, where a comprehensive understanding of both operating system theory and practical Linux applications is highly valuable.

Course Code: **CMSGCOR04T and CMSGCOR04P**
Topic Name: **Computer System Architecture**

Course Outcome: From this course students will gain a deep understanding of the organization and components of a computer system, including the CPU, memory, input/output devices, and the interconnection structure. Learn about instruction sets, addressing modes, and the design principles of the instruction set architecture. Understand how instructions are executed by the CPU. Explore the design and microarchitecture of processors, including pipelining, instruction-level parallelism, and techniques for improving CPU performance. Understand the memory hierarchy, including cache memory, main memory, and secondary storage. Learn about memory management techniques and their impact on system performance. Study the principles of input/output systems, including I/O interfaces, interrupt handling, and data transfer mechanisms between the CPU and peripherals. Explore bus systems and interconnection networks that facilitate communication between different components of a computer system. Understand the principles of parallel and distributed computing, including multi-core processors, parallel architectures, and the challenges of distributed systems. Gain hands-on experience with assembly language programming to reinforce the understanding of computer architecture concepts. Learn techniques for performance evaluation and optimization of computer systems, including benchmarking, profiling, and tuning. Stay updated with current trends and advancements in computer architecture, including emerging technologies and architectures.

Course Code: **CMSGDSE01T**
Topic Name: **Programming in JAVA**

Course Outcome: Students will understand the basic syntax, data types, and control flow structures in Java, including variables, loops, and conditional statements. Gain a solid foundation in OOP principles, including classes, objects, inheritance, polymorphism, encapsulation, and abstraction. Familiarize yourself with the Java Standard Library (Java API) and learn how to use pre-built classes and methods for common programming tasks. Understand the concept of exceptions in Java and learn how to handle errors and exceptions in a program. Explore basic GUI programming using libraries such as Swing or JavaFX. Learn to create interactive and user-friendly applications. Gain proficiency in reading from and writing to files, understanding file handling operations in Java. Learn about the Java Collections Framework, including data structures such as lists, sets, and maps. Understand how to use these structures for efficient data manipulation. Understand the basics of multithreading in Java, including creating and managing multiple threads to achieve concurrent execution. Explore networking concepts and how to implement networked applications in Java, including socket programming.

Course Code: **CMSGDSE02T**
Topic Name: **Discrete Structures**

Course Outcome: Students will understand the basics of set theory, including set operations, subsets, and set relations. Apply set theory concepts to solve problems and model real-world situations. Learn the principles of propositional logic, including logical operators, truth tables, and logical equivalences. Develop skills in constructing and analyzing logical expressions. Extend logical reasoning to predicate logic, including quantifiers and proofs. Understand how to express statements about elements in a set using predicate logic. Acquire proficiency in various proof techniques, such as direct proofs, proof by contrapositive, proof by contradiction, and mathematical induction. Explore fundamental concepts in number theory, including divisibility, prime numbers, greatest common divisors, and modular arithmetic. Understand counting principles, permutations, combinations, and the binomial theorem. Apply combinatorial concepts to solve problems related to arrangements and selections. Study basic concepts in graph theory, including vertices, edges, paths, cycles, and connectivity. Analyze and solve problems related to graph representations. Explore the concepts of relations and functions, including equivalence relations, partial orders, and injections/surjections. Understand how to represent and analyze relationships between sets. Introduce the concept of finite state machines and understand their application in modeling and solving problems related to computation. Learn the basics of formal languages, regular languages, and finite automata. Understand how automata theory is connected to the study of programming languages and computation. Introduce basic concepts of algorithmic complexity and analyze the efficiency of algorithms in terms of time and space complexity. Relate discrete structures to their applications in computer science, such as algorithm design, database management, and network design.

Course Code: **CMSGDSE03T**
Topic Name: **Software Engineering**

Course Outcome: Understand the fundamental concepts and importance of software engineering in the development life cycle. Explore the key activities involved in software engineering processes. Learn about various models of SDLC, such as Waterfall, Agile, and Iterative models. Understand the strengths and weaknesses of different development methodologies. Study the process of gathering, analyzing, and documenting software requirements. Learn how to manage and prioritize requirements throughout the project. Understand the principles of system design, including architectural design and detailed design. Learn how to create design documentation and make design decisions. Explore best practices for coding, including code readability, maintainability, and documentation. Understand how to implement designs into executable code. Learn various testing techniques and strategies to ensure software quality. Understand the importance of test planning and test documentation. Study the challenges and techniques involved in software maintenance. Understand version control and configuration management. Learn project management principles specific to software engineering. Understand how to estimate project effort, schedule tasks, and manage resources. Explore metrics for measuring software quality, productivity, and performance. Understand how to use metrics for project tracking and improvement. Identify and analyze potential risks in software projects. Learn risk mitigation and contingency planning strategies. Understand collaborative development practices, including version control systems (e.g., Git). Learn about tools for collaboration, issue tracking, and continuous integration. Explore ethical considerations in software engineering. Understand the responsibilities of software engineers in a professional context. Emphasize the importance of clear and effective communication in software development. Learn how to create and maintain project documentation.

Course Code: **CMSGDSE04T**
Topic Name: **Computer Networks**

Course Outcome: Define computer networks and understand their importance in modern computing. Explore the historical development and evolution of computer networks. Study different network topologies and their advantages and disadvantages. Understand various networking protocols and their roles in communication. Explore the functionalities of the physical and data link layers of the OSI model. Learn about error detection and correction mechanisms. Understand the role of the network layer in routing and forwarding data. Study common routing algorithms and protocols. Explore the functions of the transport layer in end-to-end communication. Learn about flow control, error recovery, and congestion control mechanisms. Examine the application layer protocols and their role in supporting network applications. Study common application layer protocols, such as HTTP, FTP, and DNS. Understand the challenges and protocols associated with wireless and mobile communication. Explore technologies like Wi-Fi, Bluetooth, and cellular networks. Learn about common security threats in computer networks. Explore security mechanisms, such as firewalls, encryption, and intrusion detection systems. Understand the principles of network management and monitoring. Analyze real-world case studies of successful network implementations. Apply theoretical knowledge to practical scenarios. Gain practical experience through hands-on labs and projects. Configure and troubleshoot network setups to reinforce theoretical concepts.

Course Name: **SKILL ENHANCEMENT COURSE**

Course Code: **CMSSSEC01M**
Topic Name: **Programming in Python**

Course Outcome: Understand the basic syntax and structure of the Python programming language. Learn how to write and execute simple Python programs. Explore various data types in Python, such as integers, floats, strings, and booleans. Understand how to declare and manipulate variables. Learn about control flow structures, including if statements, loops, and conditional statements. Understand how to use these statements to control the flow of a program. Define and use functions to modularize code. Explore function parameters, return values, and the concept of scope. Introduction to fundamental data structures in Python, including lists, tuples, sets, and dictionaries. Learn how to perform operations on these data structures. Understand how to read from and write to files using Python. Explore file input/output operations. Learn the basics of exception handling to deal with errors gracefully. Understand try, except, and finally blocks. Introduction to OOP principles, including classes and objects. Learn how to create and use classes in Python. Understand how to use Python modules and libraries to extend functionality. Explore commonly used libraries, such as NumPy for numerical computing and matplotlib for data visualization. Introduction to web development using frameworks like Flask or Django (if covered in the course). Understand the basics of creating web applications with Python. Develop problem-solving skills using Python. Understand algorithmic thinking and basic algorithm design. Introduction to version control systems, particularly Git. Learn basic Git commands for collaborative coding and project management. Emphasize good coding practices, code readability, and style conventions. Understand the importance of documentation.

Course Name: **SKILL ENHANCEMENT COURSE**

Course Code: **CMSSSEC02M**
Topic Name: **R Programming**

Course Outcome: The course in R Programming aims to impart students with a comprehensive understanding of the R programming language and its applications in statistical analysis, data visualization, and data manipulation. Throughout the course, students typically learn the fundamentals of R, including syntax, data structures, and control flow. Emphasis is placed on statistical analysis using R, enabling students to conduct hypothesis testing, regression analysis, and exploratory data analysis. The course often covers data visualization techniques using packages like ggplot2, enhancing students' abilities to communicate insights effectively. Students may also gain proficiency in data manipulation tasks with tools like the dplyr package. Additionally, the course may introduce concepts related to reproducibility and version control, fostering good coding practices. By the end of the course, students are expected to be well-equipped to use R as a powerful tool for statistical computing and data analysis in various domains, including academia, research, and industry.