Course: Bachelor of Computer Application (BCA)
Course: Duration: 3 years
(As per latest revised GU-NEP 2020)

Program Objective:

With the rapid advancement and integration of Computers and Information Technology into our daily lives, there is a growing demand for skilled professionals who can develop, maintain, and effectively utilize computing systems and software. One key factor behind the success of modern businesses is their transformation into information-enabled enterprises, where Information Technology serves not merely as an enabler, but as a critical driver of innovation and growth. IT is used extensively—from simple office automation to strategic decision support, business process re-engineering, and organizational transformation. To harness the immense capabilities of IT, there is a pressing need for a new generation of tech-savvy professionals and managers. The **Bachelor of Computer Applications (BCA)** programme, introduced by **Gauhati University** is designed to bridge this gap by producing industry-ready graduates equipped with the knowledge and skills required to thrive in the IT sector.The program covers the various essential concepts in Computer Science.

- The course lays a structured foundation of Computer fundamentals, Numerical methods, Data structure, and Algorithm and Complexity analysis, Software Engineering.

- Programming Concepts in various languages (C, C++, Java, Visual Basic etc.)
- The program covers the concept of Computer Networking, System Programming and Administration, Operating System, Digital Image Processing, Embedded systems, Computer Architecture, Microprocessor,

- It also provides detail knowledge PHP programming, Numerical methods, Combinatorial optimization, Computer Graphics and Database management system.

- An exceptionally broad range of topics covering current trends and technologies in computer science: Programming in Python, Cyber Security, Data mining, R-Programming, Data Sciences, Artificial Intelligence and Android Programming.

- Also, to carry out the hand on sessions in Computer lab using various Programming mlanguages and tools to have a deep conceptual understanding of the topics to widen the horizon of students self-experience.

Course Outcome:

FIRST SEMESTER		
PAPER CODE	**PAPER NAME**	**COURSE OUTCOME**
	Computer Fundamentals	**Upon completion of this course, students will able :** • To understand the fundamental concepts of number systems, computer components, classification, and data representation formats such as ASCII and Unicode. • To explain the types, hierarchy, and characteristics of memory and storage devices including semiconductor, magnetic, optical, and flash memory. • To identify and describe the working of various input devices such as keyboard, mouse, scanner, OMR, OCR, MICR, and digital input tools. • To understand and differentiate among output devices including types of monitors, printers, plotters, and voice output systems. • To apply the basics of programming languages, design algorithms and flowcharts, and differentiate between system software and application software. • To explain computer network concepts, internet services, and basic web development using HTML tags and CSS, along with understanding network tools like FTP and email.
	Introduction to C-Programming	**Upon completion of this course, students will :** • Able to implement the algorithms and draw flowcharts for solving Mathematical and Engineering problems. • Demonstrate an understanding of computer programming language concepts.

		• To be able to develop C programs on Linux platform. • Ability to design and develop Computer programs, analyses, and interprets the concept of pointers, declarations, initialization, operations on pointers and their usage. • Able to define data types and use them in simple data processing applications also he/she must be able to use the concept of array of structures. • Student must be able to define union and enumeration user defined data types. • Develop confidence for self-education and ability for life-long learning needed for Computer language.
	Mathematics I	**Upon completion of this course, students will be able to:** • Define the concepts of sets, relations, and functions from a Computer Science perspective. • Understand and interpret statistical and probabilistic measures commonly used in computational software and packages. • Solve problems involving metrics and determinants. • Apply basic statistical and probability techniques to solve relevant computational problems. • Utilize arrays as two-dimensional matrices in problem-solving contexts.

SECOND SEMESTER		
	Data Structures & Algorithms Using C	**Upon completion of this course, students will be able to:** • Understand basic data structures, data types, and represent arrays in memory. • Implement and perform operations on various types of linked lists. • Apply stack and queue operations for

		expression conversion and evaluation. • Represent, traverse, and manipulate binary trees and binary search trees. • Implement searching and sorting algorithms with attention to efficiency. • Analyze algorithms using time and space complexity and asymptotic notations.
CIT	Digital Logic Fundamentals	**Upon completion of this course, students will be able to:** • Understand binary number systems, perform base conversions, and apply binary codes for data representation and error detection. • Apply Boolean algebra rules and logic gate functions to design and analyze digital circuits. • Simplify Boolean functions using Karnaugh maps and tabulation methods. • Design and implement basic combinational circuits such as adders, subtractors, encoders, decoders, and multiplexers. • Understand the working of sequential circuits and design them using flip-flops, counters, and registers.
CIT	Mathematics II	**After completing this course, students will be able to:** • Understand limits, continuity, and derivatives; apply key theorems; analyze functions for maxima and minima. • Describe graph types and representations; perform basic operations and traversals like BFS and DFS. • Apply counting principles, pigeonhole principle, and solve problems using permutations and combinations. • Use logical connectives, truth tables, normal forms, and basic inference rules in propositional logic.

THIRD SEMESTER		
CIT	Computer Organization and Architecture	**After completing this course, students will be able to:** • Understand computer architecture basics, functional blocks, and types of CPU registers. • Represent and manipulate data using number systems, binary codes, and arithmetic operations. • Explain register transfer language and perform basic arithmetic, logic, and shift operations. • Understand instruction formats, addressing modes, control units, and CPU data paths. • Explain memory types, cache and virtual memory concepts, and memory hierarchy. • Describe I/O control methods and interrupt handling techniques. • Understand 8085 microprocessor architecture, instruction sets, and write simple assembly programs.
CIT	System Software	**Upon completion of this course, students will be able to:** • Understand system software types and the architecture of the Simplified Instructional Computer (SIC). • Explain assembler functions, instruction handling, and design options including NASM assembler basics. • Describe loading, relocation, and linking processes; distinguish between static and dynamic linking. • Understand macro definitions, expansion, and macro processing in language translators.

		• Explain compiler phases including parsing, code generation, and optimization; distinguish compilers and interpreters.
CIT	Object Oriented Programming through C++	**After completing this course, students will be able to:** • Understand OOP concepts, structure of C++ programs, data types, operators, and control structures. • Define and use classes, objects, constructors, destructors, arrays of objects, and friend functions. • Apply function and operator overloading, including unary, binary, and type conversion techniques. • Implement different types of inheritance, use pointers with classes, and apply runtime polymorphism. • Perform formatted and unformatted I/O operations using C++ stream classes and manipulators. • Handle file operations such as reading, writing, and random access using file stream classes.
FOURTH SEMESTER		
CIT	Database Management System	**After completing this course, students will be able to:** • Understand the basic concepts, need, components, and users of DBMS. • Explain DBMS architecture, data independence, and various data models. • Design ER diagrams using entities, relationships, and mapping constraints; convert to

		relational tables. • Understand relational data structures and apply relational algebra operations. • Analyze functional dependencies and apply normalization up to BCNF to reduce anomalies. • Explain transaction properties and concurrency control mechanisms. • Use SQL to define, manipulate, and query data, including joins and aggregate functions.
CIT	Operating system	**After completing this course, students will be able to:** • Understand the structure, types, functions, and history of operating systems, including virtualization and cloud computing. • Explain process states, process scheduling, inter-process communication, and threading models. • Analyze synchronization issues and apply software/hardware tools to manage concurrent processes. • Identify causes of deadlock and apply prevention, avoidance, detection, and recovery techniques. • Understand memory allocation strategies, virtual memory, paging, segmentation, and page replacement algorithms.
CIT	Automata Theory and Languages	**Upon completion of this course, students will be able to:** • Understand and construct

		deterministic and non-deterministic finite automata (DFA, NFA), and perform state minimization. • Explain regular languages, regular expressions, and their equivalence to finite automata and regular grammars. • Apply closure properties and decision algorithms; prove non-regularity using the pumping lemma and pigeonhole principle. • Design and simplify context-free grammars (CFGs), understand derivations and ambiguity, and apply normal forms and pumping lemma for CFLs. • Design pushdown automata and demonstrate their equivalence to context-free languages; distinguish between deterministic and non-deterministic PDA.
CIT	Python Programming	**Upon completion of this course, students will be able to:** • Understand the fundamentals of Python programming including syntax, identifiers, keywords, operators, and I/O operations. • Apply control structures (loops, conditionals) and define functions with various argument types, including lambda functions. • Use and manipulate built-in Python data structures such as lists, tuples, sets, and dictionaries. • Implement exception handling mechanisms and define user-defined exceptions in Python. • Perform file input and output operations and develop programs that read/write data from/to files

		and logs. • Demonstrate understanding of OOP concepts in Python including classes, inheritance, polymorphism, and custom exceptions. • Explore and apply essential Python libraries such as NumPy, Matplotlib, OpenCV, and Tkinter in basic applications. • Develop Python programs that connect to and interact with databases using SQL operations like INSERT, SELECT, UPDATE, DELETE, COMMIT, and ROLLBACK.
FIFTH SEMESTER		
CIT	Software Engineering	**Upon completion of this course, students will be able to:** • Understand the basic concepts of Software Engineering, distinguish it from Computer Science and Systems Engineering, and identify characteristics of good software products. • Compare and contrast different software development life cycle (SDLC) models and evaluate their advantages and disadvantages. • Analyze and document software requirements using Software Requirement Specification (SRS) and evaluate different feasibility aspects of software projects. • Apply project management principles in software development, including cost estimation techniques, project

		scheduling, and team structures. • Demonstrate knowledge of design principles like cohesion and coupling, create Data Flow Diagrams (DFDs), and understand UML modeling for Object-Oriented Design. • Apply structured coding practices and various testing methodologies including black-box and white-box techniques to ensure software quality. • Understand software reliability metrics, ISO 9000 certification, and various types of software maintenance along with cost estimation.
CIT	Web Technologies	**Upon completion of this course, students will be able to:** • Understand fundamental concepts of the Internet, WWW, and web protocols including HTTP, HTTPS, and DNS. • Develop static web pages using HTML with proper structure, formatting, and semantic elements. • Apply CSS for styling web pages, including layout, colors, typography, and the box model. • Use JavaScript to create interactive client-side web applications by manipulating the DOM and handling events. • Implement client-side form validation using JavaScript to ensure data integrity before submission. • Develop server-side web applications using PHP for form handling, file uploads, and

		database interactions. • Connect PHP applications to databases and perform basic CRUD operations with SQL queries.
CIT	Java Programming	**After completing this course, students will be able to:** • Understand the fundamentals of Java programming, its history, compilation process, and key language features. • Use Java data types, operators, control structures, and arrays effectively in program development. • Apply core object-oriented programming concepts in Java, including classes, objects, inheritance, polymorphism, and interfaces. • Perform string manipulation using Java String and StringBuffer classes and organize code with packages. • Handle exceptions and perform input/output operations using Java's built-in classes and custom exception handling. • Develop GUI applications using Swing components and event handling techniques. • Connect Java applications to databases using JDBC for data retrieval and manipulation.
CIT	Computer Networks	**Upon completion of this course, students will be able to:** • Explain the fundamental

		concepts of computer networks, including network types, topologies, protocols, and layered architectures such as the OSI and TCP/IP models. • Describe physical layer communication concepts, including signals, bandwidth, encoding schemes, modulation, multiplexing, transmission media, and switching techniques. • Understand data link layer functions, including framing methods, error detection and correction techniques, and flow control protocols. • Explain multiple access protocols and the role of network devices used in LANs and backbone networks. • Understand network layer services, routing algorithms, IP addressing, and related protocols such as ARP and RARP. • Describe transport layer services, compare TCP and UDP protocols, and explain TCP connection establishment and termination. • Summarize key application layer protocols, including DNS, WWW, email architecture, and HTTP.
SIX SEMESTER		
CIT	i) Computer Graphics	**Upon completion of this course, students will be able to:** • Understand the basic elements and applications of computer graphics. • Identify and explain various

		graphics input and output hardware devices and their working principles. • Apply fundamental graphic techniques including line drawing, circle and ellipse generation, curve generation, and area-filling algorithms. • Perform 2-D geometric transformations and understand 2-D viewing and clipping algorithms. • Understand 3-D concepts such as projections, geometric transformations, and 3-D viewing. • Explain geometric modeling concepts including Bezier and Hermite curves and surfaces. • Describe and implement visible surface determination techniques using object-space and image-space methods.
CIT	ii) Information Security and Cyber Laws	**Upon successful completion of this course, students will be able to:** • Understand fundamental concepts of computer security, including confidentiality, integrity, availability, and various cryptographic techniques. • Identify common program security threats such as buffer overflows, malicious code, and apply protection mechanisms in operating systems. • Analyze database security challenges including data integrity, access control, inference control, and apply appropriate security measures. • Explain network security threats and defenses, including firewalls,

		intrusion detection systems, encryption, and secure communication protocols. • Understand cyber laws, key sections of the Information Technology Act 2000, and their implications for combating cyber crimes and enforcing legal security.
CIT	iii) Computer Oriented Numerical and Statistical Methods	**Upon completion of this course, students will be able to:** • Understand number representations, floating-point arithmetic, and analyze errors in numerical computations. • Solve algebraic and transcendental equations using numerical methods like Bisection, Newton-Raphson, and Gauss elimination for linear systems. • Apply interpolation techniques including Newton's forward, backward, and divided difference methods for estimating data points. • Implement numerical methods to solve ordinary differential equations such as Euler's and Runge-Kutta methods. • Interpret and represent statistical data using frequency distributions, graphs, and diagrams effectively. • Calculate and analyze measures of central tendency and variation including mean, median, mode, standard deviation, and variance. • Understand probability theory concepts and apply discrete probability distributions, conditional probability, regression, and correlation

		analysis.
CIT	i) Artificial Intelligence	**Upon completion of this course, students will be able to:** • Understand the fundamentals of Artificial Intelligence, intelligent agents, and their applications. • Apply various problem-solving and search techniques including BFS, DFS, heuristic search, A* algorithm, and game playing strategies like Min-Max and Alpha-Beta pruning. • Represent knowledge using first-order logic, semantic nets, frames, scripts, production rules, and work with logic programming in PROLOG. • Handle uncertainty in AI through truth maintenance systems, default reasoning, and Bayesian probabilistic inference. • Understand the basics of natural language processing, including parsing techniques and grammar formalisms.
CIT	ii) Advanced Web Programming	**Upon completion of this course, students will be able to:** • Utilize advanced HTML5 features including semantic elements, form validation, multimedia integration, and meta information for improved SEO and user experience. • Design responsive, well-structured web layouts using advanced CSS techniques including flexbox, grid, media queries, and CSS frameworks like Bootstrap and Tailwind. • Implement interactive client-side

		functionalities using advanced JavaScript concepts such as event handling, asynchronous programming, DOM manipulation, and AJAX. • Develop server-side web applications using PHP, manage user sessions, perform CRUD operations with MySQL, and understand security concerns like XSS. • Understand web hosting options, web APIs, version control, and basics of modern web development workflows including CI/CD, security protocols, and web analytics.
CIT	iii) Data Mining and Warehousing	**Upon successful completion of this course, students will be able to:** • Understand fundamental concepts of data mining and the knowledge discovery process, including data types, tasks, and preprocessing techniques. • Describe the architecture and components of data warehousing, apply multidimensional data models, and perform OLAP operations effectively. • Analyze association rules and implement algorithms like Apriori for mining frequent itemsets in market basket data. • Apply various clustering techniques, understand distance and similarity measures, and distinguish between partitional, hierarchical, and density-based clustering methods. • Understand classification concepts, including K-Nearest Neighbor, Bayesian classifiers, and decision tree approaches.

		• Gain awareness of recent advancements in data mining such as web mining, spatial and temporal data mining, big data, neural networks, and genetic algorithms.
CIT	i) Optimization Techniques	**Upon completion of this course, students will be able to:** • Understand the fundamental concepts and classifications of optimization problems and appreciate the role of simulation and modeling techniques. • Formulate and solve linear programming problems using simplex methods, dual simplex, sensitivity analysis, and apply models such as transportation and network optimization including shortest path and minimal spanning tree algorithms. • Analyze queuing systems by understanding their elements, pure birth-death models, and advanced queuing scenarios involving arrivals and departures. • Apply unconstrained optimization techniques such as Newton, Quasi-Newton, conjugate gradient, line search, and trust region methods to solve nonlinear optimization problems. • Understand and implement constrained optimization methods including barrier and augmented Lagrangian methods, sequential quadratic programming, interior-point methods, and goal programming with its formulations and algorithms.

CIT	ii) Mobile Application Development	By the end of this course, students will be able to: • Understand mobile app development basics and compare native and cross-platform approaches. • Explain Android OS architecture and development tools. • Build native Android apps with UI design, activity lifecycle, and Material Design. • Use device features, manage permissions, and integrate multimedia and web services. • Develop database-driven apps using SQLite and Firebase. • Create cross-platform apps with Flutter and publish them on Android and iOS.
CIT	iii) Graph Theory	**At the end of the course, students will be able to:** • **Understand and apply fundamental concepts of graph theory**, including types of graphs, graph representations, and traversal algorithms such as BFS and DFS. • **Analyze connectivity and path-related problems** in graphs, including Hamiltonian and Eulerian paths and cycles, and implement classical shortest path algorithms like Dijkstra's, Bellman-Ford, and Floyd-Warshall. • **Explain the properties and applications of trees**, including spanning trees, minimum spanning trees, and various tree data structures such as binary trees, AVL trees, and B-trees. • **Apply matching and coloring**

		theories to solve real-world problems using algorithms such as Ford-Fulkerson, Hungarian method, and understand concepts like vertex cover, independent sets, and chromatic numbers. • **Analyze directed graphs and flow networks**, understanding concepts like strongly connected components, tournaments, Markov chains, and implement flow algorithms such as Ford-Fulkerson. • **Solve classical graph problems** like Travelling Salesman Problem, Chinese Postman Problem, and Huffman coding, demonstrating problem-solving skills in graph optimization and coding theory.
CIT	Project	• At the end of the course, students will be able to demonstrate the ability to identify real-world problems, design effective solutions using appropriate technologies, and present their work in a structured technical report.

Note: Paper Code is to updated as per GU directives.