SYLLABUS FOR BCA (MAJOR)

Under Single Major Single Minor (FYUGP)

(To be implemented from the session 2024-25)

Sem III & IV

Proposed Syllabus for four years B.C.A. (Major) Programme

Proposed Syllabus for BCA (Major)Program Single Major Single Minor								
Year	**Semester**	**Paper Code**	**Paper**	**Credits**	**Periods/Weeks**	**Exam Marks**	**Practical/Tutorial**	**Total**
2nd Year	**3**	**BCAPMAJ305**	Object Oriented Programming with C++	3	3	60	-	**80**
		BCAPMAJ305 (Lab)	Object Oriented Programming with C++	1	1	-	20	
		BCAPMAJ306	Computer System Architecture	3	3	60	-	**80**
		BCAPMAJ305 (T)	Computer System Architecture (Tutorial)	1	1	-	20	
	4	**BCAPMAJ407**	Discrete Mathematics	3	3	60	-	**80**
		BCAPMAJ407 (T)	Discrete Mathematics (Tutorial)	1	1	-	20	
		BCAPMAJ408	Data Structure	3	3	60	-	**80**
		BCAPMAJ408 (Lab)	Data Structure (Lab)	1	1	-	20	

Detailed Syllabus

2nd Year: Semester III

Course Code: BCAPMAJ305

Course Name: Object Oriented Programming with C++

Prerequisite(s) and/or Note(s):

1. Basic knowledge of programming logic at the high school level.
2. Prior exposure to any programming language like C is helpful but not mandatory.
3. **Note(s):**
   - The syllabus is subject to minor modifications depending on institutional or academic requirements.
   - Students will be assessed only on the basis of topics covered in class during the semester.

Course Objectives

Knowledge Acquired:

1. Understand the fundamental principles of object-oriented programming (OOP).
2. Learn the syntax and semantics of the C++ programming language.
3. Develop a conceptual understanding of classes, objects, inheritance, polymorphism, and file handling.
4. Learn how to implement modular, reusable, and maintainable code using OOP concepts.

Skills Gained:

1. Ability to write efficient, error-free C++ programs using object-oriented techniques.
2. Application of arrays, functions, classes, and inheritance to solve real-world problems.
3. Skills in operator overloading, polymorphism, and working with pointers and memory management.
4. Ability to use C++ constructs to design small-scale systems or components.

Competency Developed:

1. Ability to break down a problem into objects and define appropriate class structures.
2. Capability to handle complexity through encapsulation and modularity.
3. Confidences in handling object lifecycle (construction/destruction), memory allocation, and file handling.
4. Readiness for further studies in data structures, algorithms, and system-level programming using C++ or similar languages.

BCAPMAJ305: Object Oriented Programming with C++ [Credits: 3, Lectures:45]

UNIT – I: Language Fundamental (10 Lectures)

Overview of OOP: The Object Oriented paradigm, Basic concepts of OOP, Benefits of OOP, Object oriented languages, Application of OOP

Overview of C++: Evolution of C++ from C, Structure of a C++ program Data Types: Built-in data types, User-defined data types, Derived data types, Keywords, Constants and Variables: symbolic constants, Dynamic initialization of variable, Reference variable. Operators and expressions, Input and output using cin, cout, Control Structures: if-else, nested if-else, while, do-while, for, break, continue, switch, goto statement.

UNIT – II: Arrays, Structure & Functions (10 Lectures)

Arrays: Single and multi-dimensional arrays, Character arrays (C-style strings), Introduction to C++ String class ; **Structures**: A Simple structure, defining a structure variable, Accessing structure's member, Enumeration data type ; **Function**: Function Declaration, Calling Function, Function Definition, Passing Arguments to function: Passing Constant, Passing Value, Reference Argument, Structure as argument, Default Argument. Returning values from function: return statement, Returning structure variable, Return by reference, Overloaded Function, Inline Function.

UNIT – Ill: Object and Classes (10 Lectures)

Object and Class: Defining the class and its member, Making an outside function inline, nesting of member function, array as class member, Difference between classes and structures.

Memory allocation: Memory allocation for objects, new and delete operator, static data member, static member functions, object as function argument.

Constructor & Destructor: Constructors and types of constructors (Default constructor, Copy Constructor, Parameterized constructor), Destructors.

UNIT – IV: Pointers and Inheritance (7 Lectures)

Pointers: Introduction, & and * operator, pointer to object, this pointer, pointer to derived class. **Inheritance**: Inheritance- definition, concept, Types of Inheritance, Visibility modes- Public, Private, Protected, Virtual Base Class, Benefits of Inheritance.

UNIT – V: Polymorphism (8 Lectures)

Static Polymorphism: Operator overloading, Operator keyword, overloading unary operators (++ (pre increment and post increment), –) using operator function, overloading binary operators (+, -, =, >=, <=, +=, <, >, []). Friend function, Friend class, overloading binary operators using friend function and without using friend function. **Dynamic polymorphism**: Virtual function, Pure Virtual Function, Abstract class.

Suggested Readings:

1. Object Oriented Programming with C++ : E. Balagurusamy, The McGraw-Hill
2. Let Us C++: Yashwant Kanetkar, BPB Publications
3. The C++ Programming Language: Bjarne Stroustrup, Addision Wasley.
4. Object Oriented Programming in C++ : Robert Lafore, Galgotia Publications.

BCAPMAJ305 (Lab): Object Oriented Programming with C++ [Credits:1, Lab hours-30/Week-2]

Programming in C++ (Lab)

1. Write a program to display your name, age, and course using cout.
2. Write a program to calculate the area of a rectangle using user input.
3. Write a program to check whether a number is even or odd using if-else.
4. Write a program to find the largest of three numbers using nested if-else.
5. Write a program to print the multiplication table of a number using a for loop.
6. Write a program to find the sum of digits of a number using a while loop.
7. Write a program to find the maximum and minimum elements in an array.
8. Write a program to reverse an array.
9. Write a program to sort an array using bubble sort.
10. Write a program to check whether a string is a palindrome (use character array).
11. Write a program using a structure to store and display student details (name, roll, marks).
12. Write a function to find the factorial of a number using recursion.
13. Write a program to overload a function sum() to calculate sum of 2, 3, or 4 integers.
14. Write a program that passes a structure as a function argument and displays data.
15. Write a program to create a class Rectangle with data members length and breadth, and member functions to calculate area and perimeter.
16. Write a program to create a class Student with constructor and destructor that displays a message when invoked.
17. Write a program to demonstrate use of static data members and static functions in a class.
18. Write a program to pass an object as an argument to a function.
19. Write a program to demonstrate the use of pointers with variables and arrays.
20. Write a program to create a base class Person and a derived class Student. Use inheritance to input and display details.
21. Write a program to implement multilevel inheritance (e.g., Person → Employee → Manager).
22. Write a program to demonstrate use of this pointer.
23. Write a program to overload the + operator to add two complex numbers.
24. Write a program using virtual functions to demonstrate runtime polymorphism.
25. Write a program to create an abstract class Shape with a pure virtual function area(), and derive Circle and Rectangle from it.

2nd Year: Semester III

Course Code: **BCAPMAJ306**
Course Name: Computer System Architecture

Brief Course Description:

The course computer system architecture course aims to describe a broad range of architectural designs and to contrast them, highlighting the design decisions they incorporate, and how these design decisions impact program performance.

Prerequisite(s) and/or Note(s):

1. Students who have not taken Digital Systems will need to do additional background reading on combinational circuits and assembler programming.

Course Objectives:

Knowledge acquired:

1. Understanding computer architecture and organization is essential for computer scientists, engineers, and programmers to develop efficient and high-performance systems.
2. It forms the foundation for designing and optimizing hardware and software.

Skills gained:

1. Understanding System Performance: It enables individuals to analyze and optimize the performance of computer systems by understanding how hardware components interact and function.
2. Efficient Programming Skills: Knowledge of architecture helps in writing more efficient and optimized code by understanding how instructions are executed at the hardware level.
3. Hardware-Software Integration: It provides insight into the coordination between hardware and software, which is essential for system-level programming and embedded systems design.
4. Problem Solving and Debugging: A clear grasp of internal system operations aids in effective troubleshooting, debugging, and diagnosing hardware-related software issues.
5. Foundation for Advanced Fields: It lays the groundwork for advanced topics like operating systems, compilers, and computer networks, enabling deeper understanding and innovation in these areas.

Competency Developed:

1. Architectural Design Skills: Enables individuals to understand and contribute to the design and development of computer hardware and system architecture.
2. Embedded Systems Expertise: Equips learners with the foundational knowledge required to design and program embedded systems used in devices like smartphones, IoT devices, and industrial controllers.
3. System Optimization Abilities: Develops the skills to analyze, evaluate, and enhance the performance and efficiency of computing systems at both hardware and software levels.
4. Research and Innovation Capability: Prepares individuals to engage in cutting-edge research in fields like quantum computing, parallel processing, and high-performance computing technologies.
5. Comprehensive System Understanding: Fosters a holistic grasp of how computer systems operate, enabling effective collaboration across software, hardware, and systems engineering domains.

2nd Year: Semester III

BCAPMAJ306: Computer System Architecture [Credits:3, Lectures:45]

Unit 1: Register transfer and microoperations (6 lectures)

Register transfer, Arithmetic microoperations, logic microoperations, shift microoperations, Computer registers, Need of Registers, bus system, Interconnection Structures, Bus Interconnection.

Unit 2: Basic Computer Organization and Design (7 lectures)

Instruction set, timing and control, instruction cycle, hardwired instruction format, memory reference, register reference and input-output instructions. Design of basic computer, Interrupt.

Unit3: Central Processing Unit (12 lectures)

Stack organization, microprogrammed control. Microprogrammed instruction formats, addressing modes, instruction codes, machine language, assembly language, design of CPU, RISC, CISC architectures.

Unit4: Computer Arithmetic (10 lectures)

Addition and Subtraction, Multiplication algorithm (Boots), Division algorithm, Floating point arithmetic operations.

Unit5: Memory and Input-Output Organization (10 lectures)

Memory hierarchy, Main memory, Cache memory, Virtual memory. Input/output: External Devices, I/O Modules, Programmed I/O, Interrupt-Driven I/O, Direct Memory Access, I/O Channels.

Suggested Readings:

1. M. Mano, Computer System Architecture, Pearson Education 1992
2. A. J. Dos Reis, Assembly Language and Computer Architecture using C++ and JAVA, Course Technology,2004
3. W. Stallings, Computer Organization and Architecture Designing for Performance, 8thEdition, Prentice Hall of India,2009
4. M. M. Mano, Digital Design, Pearson Education Asia,2013
5. Carl Hamacher, Computer Organization, Fifth edition, McGraw Hill, 2012.

BCAPMAJ306(T): Computer System Architecture	**[Credits: 1, Lecture Hours:15]**

Computer System Architecture tutorials as assigned and advised by teacher(s).

Detailed Syllabus

2nd Year: Semester IV

Course Code: BCAPMAJ407

Course Name: Discrete Mathematics

Brief Course Description:

This paper deals with the basic idea about discrete mathematics and discrete structures, about mathematical notations and their use.

Prerequisite(s) and/or Note(s):

1. High school mathematics.
2. Note(s): Syllabus changes yearly and may be modified during the term itself, depending on the circumstances. However, students will be evaluated only on the basis of topics covered in the course.

Course Objectives:

Knowledge acquired:

1. Basic knowledge of discrete mathematics and discrete structures,
2. To develop understanding of Logic sets and functions
3. Knowledge of mathematically correct terminology and notations.
4. Knowledge about construction of direct and indirect proofs..

Skills gained:

(1) Development of problem-solving skills necessary for understanding counting problems.
(2) Ability to generalize from a single instance of a problem an entire class of problems and identification of patterns of data.

Competency Developed:

1. Ability to analyze problems and solve problems.
2. Ability to implement mathematical knowledge in data analysis.

BCAPMAJ407: Discrete Mathematics [Credits: 3, Lectures:45]

Unit 1: Set Theory (8 Lectures)

Introduction, Set properties, Venn diagram, subsets, Combination of Sets; union, intersection, difference of sets, set complements, disjoint sets, power set, Multi-sets, Ordered Pairs, Cartesian Products, Set Identities

Unit 2: Relations and Functions (10 Lectures)

Introduction, Binary Relations, Operations on Relation, Domain and Range of Relation, Properties of Relations; Equivalence Relations, Equivalence classes, Partition of Set, Representation of Relation, Composite relations, Order of relations, Classification of Functions, Floor and Ceiling Functions, Operations on Functions

Unit 3: Natural Numbers (7 Lectures)

Mathematical induction, proof format, Fibonacci identity, Binomial distribution, variants of induction; strong induction

Unit 4: Propositional Logic (7 Lectures)

Introduction, Sentences, Statements, Well formed formula, truth table, complete truth tables, tautology, Logical equivalence, Theory of inference; formula representation, Rules of inference, Rules of conditional proof, Rules of indirect proof

Unit 5: Recurrence Relations (6 Lectures)

Linear Recurrences, Non-homogeneous Recurrences, Growth of functions, Big-O notation, Big – Ω notation, Big-Θ notation, properties of asymptotic orders

Unit 6: Combinatory (7 Lectures)

Permutations, permutations with repetitions, circular permutations, Combination, combination with repetitions, Principal of Inclusion-Exclusion, Pigeonhole Principle.

Suggested Readings:

1. *YN Singh (2010),**Discrete Mathematical Structures*** –Wiley.

2. *Pal and Das(2010**),*** **BCA MATHEMATICS VOLUME-I** – U. N. Dhur& Sons Pvt. Ltd. (2nd edition)

3. *Liu and Mohapatra(2008) ,**Elements of Discrete Mathematics*** – McGraw Hill Education

4. *Liu and Mohapatra*(2017),**Elements of Discrete Mathematics: A Computer Oriented Approach**, McGraw Hill Education; 4th edition

BCAPMAJ407(T): Discrete Mathematics Tutorial[Credits:1,Lecture Hours:15]

Discrete mathematics tutorials as assigned and advised by teacher (s)

Detailed Syllabus

2nd Year: Semester IV

Course Code: BCAPMAJ408

Course Name: Data Structure

Brief Course Description:

This course introduces students to the fundamental concepts of data structures and their implementation using the C programming language. Emphasizing theoretical understanding and practical application, the course covers the design, analysis, and implementation of essential data structures that form the building blocks of efficient algorithms.

Prerequisite(s) and/or Note(s):

Basic knowledge of C programming language and familiarity with fundamental programming concepts.

Course Objectives:

Knowledge acquired:

Upon completing the course, students should be able to design, implement, and analyze the performance of fundamental data structures using the C programming language. They will gain problem-solving skills and be prepared for more advanced coursework in algorithms and software development.

Skills gained:

Studying data structures provides individuals with a variety of skills that are valuable in computer science, software engineering, and other related fields. Here are key skills gained from learning about data structures: algorithmic thinking, problem solving, analytical skills, critical thinking, abstraction and modeling etc. These skills collectively contribute to a well-rounded foundation in computer science and software engineering, making individuals more proficient in developing efficient and scalable software solutions.

Competency Developed:

Studying data structures helps individuals develop a range of competencies that are essential in computer science, software engineering, and related fields. Here are key competencies developed from learning about data structures: efficient resource utilization, memory management, debugging proficiency, adaptability, collaboration and teamwork, documentation, continuous learning etc. These competencies collectively prepare individuals for roles in software development, algorithm design, system optimization, and other areas where a deep understanding of data structures is crucial. They contribute to the ability to create efficient, scalable, and robust software solutions.

Syllabus Overview

Unit 1:	Introduction and Arrays	4 Lectures

Definition of data structure, need, operations on data structures. Single and Multi-dimensional Arrays, Insertion, and deletion operations.

Unit 2:	Linked Lists	10 Lectures

Singly, Doubly and Circular Lists; Normal and Circular representation of Stack in Lists. Insertion and deletion operations on singly, doubly and circular lists.

Unit 3: Stacks and Queues **12 Lectures**

Implementing single/multiple stack/s in an Array, (Array and Linked representation), Push, Pop operations, Prefix, Infix and Postfix expressions, Applications of stack; Evaluation of theses expressions, conversion of these Expressions from one to another, Limitations of Array representation of stack. Array and Linked representation of Queue, De-queue, Priority Queues, Operations of queue.

Unit 4: Trees **12 Lectures**

Introduction to Tree as a data structure; General tree, Binary Trees, conversion of a general tree to a binary tree, traversal of a binary tree, Binary search tree, AVL tree.

Unit 5: Sorting and Searching **7 Lectures**

Linear Search, Binary Search, Comparison of Linear and Binary Search, Selection Sort, Bubble Sort, Insertion Sort, Comparison of Sorting Techniques

Suggested Readings

1. Seymour Lipschutz, "Data Structures with C", Tata McGraw Hill Education Pvt Ltd.
2. Yashvant Kanetkar, "Data structures through C", bpb
3. Deshpande Kakde, "C and Data structures", dreamtech
4. Aaron M. Tenenbaum, Moshe J. Augenstein, Yedidyah Langsam, "Data Structures Using C and C++",2nd Edition, PHI, 2009.
5. Robert L.Kruse, "Data Structures and Program Design in C++", Pearson,1999.
6. D.S Malik, Data Structure using C++, 2nd Edition, Cengage Learning,2010.

Course-MAJOR Paper: **Paper Code- BCAPMAJ408L Data Structure (Lab)** **Credits-1** **Lab hours/Week-2**

Students are advised to do laboratory/practical practice not limited to, but including the following types of problems in C or C++:

1. Write a program to search an element from a list. Give user the option to perform Linear or Binary search.
2. WAP to sort a list of elements. Give user the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists.
4. Implement Doubly Linked List. Include functions for insertion, deletion and search of a number.
5. Implement Circular Linked List. Include functions for insertion, deletion and search of a number.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation.
8. Perform Queues operations using Circular Array implementation.
9. Create and perform different operations on Double-ended Queues using Linked List implementation and array implementation.
10. WAP to calculate factorial and to compute the factors of a given no. (i)using recursion, (ii) using iteration
11. WAP to display Fibonacci series (i) using recursion,(ii)using iteration
12. WAP to reverse the order of the elements in the stack using additional stack.
13. WAP to reverse the order of the elements in the stack using additional Queue.
14. WAP to implement Diagonal Matrix using one-dimensional array.
15. WAP to implement Lower Triangular Matrix using one-dimensional array.
16. WAP to implement Upper Triangular Matrix using one-dimensional array.
17. WAP to implement Symmetric Matrix using one-dimensional array.