Syllabus
For

4 years Undergraduate Programme
of
Bachelor of Computer Application
(BCA)

(as per NEP 2020 guideline)

NORTH-EASTERN HILL UNIVERSITY
MEGHALAYA

Course Structure

1st Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA - 100	**MAJOR**: Problem Solving and Programming in C	3	0	2	56	19	75	25	100	3	1	4
BCA - 100	**MINOR**: Problem Solving and Programming in C	3	0	2	56	19	75	25	100	3	1	4
SEC - 130	**SEC:** Cyber Security *(Existing)*	2	0	2	40	16	56	19	75	2	1	3
MDC -112	**MDC:** Fundamentals of Computer Systems *(Existing)*	3	0	0	56	0	56	19	75	3	0	3
	AEC:											3
	VAC:											3
Total												**20**

2nd Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA - 150	**MAJOR**: Web development using PHP and MySQL, Server Side Frameworks	3	0	2	56	19	75	25	100	3	1	4
BCA - 150	**MINOR**: Web development using PHP and MySQL, Server Side Frameworks	3	0	2	56	19	75	25	100	3	1	4
SEC - 183	**SEC:** Python Programming (Existing)	2	0	2	40	16	56	19	75	2	1	3
	MDC:								75			3
	AEC:								75			3
	VAC:								75			3
Total												**20**
● To exit with **UG Certificate**, students have to complete **40 credits** and also an internship/vocational/apprenticeship/community and service/field-based learning/minor project of **4 credits**												
Internship* (120 hours)							**75**	**25**	**100**			**4**

3rd Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA - 200	**MAJOR**:Object Oriented Programming using C++	3	0	2	56	19	75	25	100	3	1	4
BCA - 201	**MAJOR**: Digital Logic and Computer Fundamentals	4	0	0	75	0	75	25	100	4	0	4
	SEC:											3
	MDC:											3
	AEC:											2
	VTC:											4
Total												**20**

4th Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA - 250	**MAJOR**: Data Structure using C	3	0	2	56	19	75	25	100	3	1	4
BCA - 251	**MAJOR**: Computer System Architecture	4	0	0	75	0	75	25	100	4	0	4
BCA - 252	**MAJOR**: Database Management Systems	3	0	2	56	19	75	25	100	3	1	4
BCA - 253	**MAJOR**: Operating System and Shell Programming	3	0	2	56	19	75	25	100	3	1	4
	VTC:								100			4
Total												**20**
● To exit with **UG Diploma**, students have to complete **80 credits** and also an internship/vocational/apprenticeship/community and service/field-based learning/minor project of **4 credits**												
Internship* (120 hours)							**75**	**25**	**100**			**4**

5th Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA-300	**MAJOR**: Software Engineering	4	0	0	75	0	75	25	100	4	0	4
BCA-301	**MAJOR**: Mathematics I	3	0	2	56	19	75	25	100	3	1	4
BCA-302	**MAJOR:** Advanced Python Programming	3	0	2	56	19	75	25	100	3	1	4
BCA-302	**MINOR:** Advanced Python Programming	3	0	2	56	19	75	25	100	3	1	4
BCA-303	**INTERNSHIP (120 hours)**				0	75	75	25	100	0	4	4
Total												**20**

6th Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA-350	**MAJOR**: Mathematics II	4	0	0	75	0	75	25	100	4	0	4
BCA-351	**MAJOR**: Data Communications and Networking	4	0	0	75	0	75	25	100	4	0	4
BCA-352	**MAJOR**: Object Oriented Programming in JAVA	3	0	2	56	19	75	25	100	3	1	4
BCA-353	**MAJOR:** Data Mining	3	0	2	56	19	75	25	100	3	1	4
	VTC:								100			4
Total												**20**

7th Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA - 400	**MAJOR**: Research Methodology and Proposal Writing	4	0	0	75	0	75	25	100	4	0	4
BCA - 401	**MAJOR**: Text Analytics	3	0	2	56	19	75	25	100	3	1	4
BCA - 402	**MAJOR**: Theory of Computation and Compiler Design	4	0	0	75	0	75	25	100	4	0	4
BCA - 403	**MAJOR:** Internet of Things	3	0	2	56	19	75	25	100	3	1	4
BCA - 404	**MINOR:** Internet of Things	3	0	2	56	19	75	25	100	3	1	4
Total												**20**

8th Semester												
Sub Code	**Subject Name**	**L**	**T**	**P**	**End Sem Exam**			**IA**	**Total**	**Credits**		
					Th	**Pr**	**T**			**Th**	**Pr**	**T**
BCA-450	**MAJOR**: Machine Learning	3	0	2	56	19	75	25	100	3	1	4
BCA-451	**MINOR**: Machine Learning	3	0	2	56	19	75	25	100	3	1	4
BCA-452	**Research Project/Dissertation** ***(For Students eligible for Honours with Research) ****											12
BCA-453	**MAJOR**: Computer Oriented Numerical Methods	3	0	2	56	19	75	25	100	3	1	4
BCA-454	**MAJOR**: Mobile Application Development	3	0	2	56	19	75	25	100	3	1	4
BCA-455	**MAJOR**: Artificial Intelligence	3	0	2	56	19	75	25	100	3	1	4
Total												**20**

Prepared according to NEHU guidelines communicated to Principals of all affiliated Colleges, by CDC via letter *No CDC/NEP2020/2022/-1818* dated *15th September 2023*

In line with:

- Ordinance OC-8 (2023) as adopted in the 110th Academic Council
- Regulation RC-12 (2023) as adopted in the 110th Academic Council

** Explanation: Students securing at least 75% aggregate marks on completion of third academic year are eligible for Honours with Research of a 4-year Bachelor's Degree Programme. Those securing less than 75% are eligible for joining Honours only. Hence BCA-453 is of research nature (12 credits) while in BCA-454 (4 credits) students can apply concepts (such as software engineering) learnt in the previous semesters to develop a working piece of software.*

Students eligible for Honours with Research shall offer BCA-452 in lieu of BCA-453, BCA-454 and BCA-455

Semester wise Course Content of the 4 years BCA Syllabus

Semester III

Object Oriented Programming using C++

(MAJOR : T + P)
BCA-200

Paper Title: Object Oriented Programming using C++
Paper Code: BCA-200.b
Number of Hours Per Week : (L+T+P = 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** To inculcate knowledge on the key principles of Object-Oriented Programming.
- **CO2:** Understand and implement Object Oriented Programming constructs using C++.
- **CO3:** Proficiency in C++ programming using standard C++ libraries and debugging tools to develop robust applications.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Understand Core Object Oriented Concepts such as encapsulation, inheritance, and polymorphism, and apply them to design and implement C++ programs.
- **LO2:** Create well-designed and modular classes that encapsulate data and behaviour, promoting code reusability, maintainability, and scalability in C++ applications.
- **LO3:** Effectively apply inheritance and polymorphism to design and implement class hierarchies, promoting code reuse and flexibility in handling diverse types of objects.
- **LO4:** Implement proper memory management techniques in C++ programs, including dynamic memory allocation and deallocation, to avoid memory leaks and ensure efficient resource utilisation.

Outline of the Course:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Introduction to C++: key concepts of Object-Oriented Programming, control structures , functions	15	18	19
II	Classes and Objects, Function overloading and Operator overloading	15	19	
III	Inheritance, Pointers, Polymorphism, Exception Handling	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Introduction to C++: key concepts of Object-Oriented Programming, Advantages of Object Oriented Languages, I/O in C++, C++ Basic data types, User defined data types, Variable Declaration, Dynamic initialization, Reference variables, Operators in C++: scope resolution operator, memory management operators, typecast operator, Control Structures: Decision Making and Statements : If ..else, jump, goto, break, continue, Switch case statements, Loops in C++ : for, while, do - functions in C++.

UNIT II **15 Hours**

Classes and Objects: Declaring Objects, Defining Member Functions, Static data members and functions, Array of objects, Friend functions, Constructor and Destructors, Operator Overloading: Overloading unary and binary operators, Overloading with Friend functions, Type conversion.

UNIT III **15 Hours**

Inheritance: Types of Inheritance – Single, Multilevel, Multiple, Hierarchal, Hybrid, Multipath inheritance ,Virtual base Classes, Abstract Classes, Constructors in derived classes , Pointers: Pointer to object ,Compile time and Runtime Polymorphism, this pointer, Pointers to derived classes, Virtual functions, Exception Handling: Basics of Exception Handling – throwing mechanism, catching mechanism, rethrowing an exception.

UNIT IV **30 Hours**

Practicals from UNIT I, UNIT II and UNIT III

Suggested Experiments:

1. Write a temperature-conversion program that gives the user the option of converting Fahrenheit to Celsius or Celsius to Fahrenheit. Then carry out the conversion. Use floating-point numbers. Interaction with the program might look like this:

   Type 1 to convert Fahrenheit to Celsius,
   2 to convert Celsius to Fahrenheit: 1
   Enter temperature in Fahrenheit: 70
   In Celsius that's 21.111111
2. Write a program in C++ to find the Greatest Common Divisor (GCD) of two numbers
3. Write a C++ program that asks the user to enter positive integers in order to process count, maximum, minimum, and average or terminate the process with -1.
4. Write a program in C++ to display the sum of the series [1+x+x^2/2!+x^3/3!+....].
5. Write a program in C++ to display the pattern like a right angle triangle with numbers.

   Sample Input number of rows: 5 Output:

1
12
123
1234
12345

6. Create a four-function calculator for fractions. (See Exercise 9 in Chapter 2,and Exercise 4 in this chapter.) Here are the formulas for the four arithmetic operations applied to fractions:
   Addition: a/b + c/d = (a*d + b*c) / (b*d)
   Subtraction: a/b - c/d = (a*d - b*c) / (b*d)
   Multiplication: a/b * c/d = (a*c) / (b*d)
   Division: a/b / c/d = (a*d) / (b*c)
   The user should type the first fraction,anoperator,and a second fraction. The program should then display the result and ask whether the user wants to continue.
7. Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called power() that takes a double value for nand an int value for p,and returns the result as a doublevalue. Use a default argument of 2 for p, so that if this argument is omitted, the number n will be squared. Write a main() function that gets values from the user to test this function
8. Write a function called hms_to_secs()that takes three int values—for hours, minutes, and seconds—as arguments,and returns the equivalent time in seconds (type long). Create a program that exercises this function by repeatedly obtaining a time value in hours, minutes and seconds from the user (format 12:59:59), calling the function and displaying the value of seconds it returns.
9. Write a function called swap()that interchanges two int values passed to it by the calling program. (Note that this function swaps the values of the variables in the calling program,not those in the function.) You'll need to decide how to pass the arguments. Create a main() program to exercise the function.
10. Create a class called timedate has separate int member data for hours, minutes, and seconds. One constructor should initialise this data to 0, and another should initialise it to fixed values. Another member function should display it, in 11:59:59 format. The final member function should add two objects of type time passed as arguments. A main() program should create two initialised time objects and one that isn't initialised. Then it should add the two initialised values together, leaving the result in the third time variable. Finally it should display the value of this third variable.
11. Create an employee class where the member data should comprise an int for storing the employee number and a float for storing the employee's compensation. Member functions should allow the user to enter this data and display it. Write a main() that allows the user to enter data for three employees and display it.
12. Extend the employee class of Exercise 14 to include a date and an etypeenum. An object of the date class should be used to hold the date of first employment; that is, the date when the employee was hired. The etype variable should hold the employee's type: labourer, secretary, manager and so on. These two items will be private member data in the employee definition, just like the employee number and salary. You'll need to extend the getemploy() and putemploy() functions to obtain this new information from the user and display it. Write a main() program that allows the user to enter data for three employee variables and then displays this data
13. Write a C++ program to sort an array of elements

14. Write a C++ program to implement a class called Circle that has private member variables for radius. Include member functions to calculate the circle's area and circumference.
15. Write a C++ program to create a class called Rectangle that has private member variables for length and width. Implement member functions to calculate the rectangle's area and perimeter.
16. Write a C++ program to create a class called Person that has private member variables for name, age and country. Implement member functions to set and get the values of these variables.
17. Write a C++ program to implement a class called BankAccount that has private member variables for account number and balance. Include member functions to deposit and withdraw money from the account.
18. Write a C++ program to create a class called Triangle that has private member variables for the lengths of its three sides. Implement member functions to determine if the triangle is equilateral, isosceles, or scalene.
19. A bookshop maintains the inventory of books that are being sold at the shop. The list includes details such as author, title, price, publisher and stock position. Whenever a customer wants a book, the sales person inputs the title and author and the system searches the list and displays whether it's available or not. If it is not, an appropriate message is displayed. If it is, then the system displays the book details and requests for the number of copies required. If the requested copies are available, the total cost of the requested copies is displayed; otherwise the message "Required copies not in stock" is displayed.
20. Design a system using a class called books with suitable member functions and constructors. Use new operator in constructors to allocate memory space required
21. Write a C++ program to implement a class called Date that has private member variables for day, month, and year. Include member functions to set and get these variables, as well as to validate if the date is valid.
22. Write a C++ program to implement a class called Student that has private member variables for name, class, roll number, and marks. Include member functions to calculate the grade based on the marks and display the student's information.
23. Write a C++ program to implement a class called Shape with virtual member functions for calculating area and perimeter. Derive classes such as Circle, Rectangle, and Triangle from the Shape class and override virtual functions accordingly.
24. Write a C++ program to dynamically create an object of a class using the new operator.
25. Write a program with the following:
    a. A function to read two double type numbers from the keyboard
    b. A function to calculate the division of these two numbers
    c. A try block to throw an exception when a wrong type of data is keyed in
    d. A try block to detect and throw an exception if the condition *"divide by zero"* occurs
    e. Appropriate catch blocks to handle the exceptions thrown

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS

	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc.)

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. E. Balagurusamy, "*Object-Oriented Programming with C++",* Tata Mc Graw Hill, 8th Edition, 2020.

2. H. Schildt, *"C++: The Complete Reference"*, McGraw Hill Education, 4th Edition, 2017

Reference Books:

1. A. N. Kamthane, "*Object-Oriented Programming with ANSI and Turbo C++",* Pearson Education, Singapore, 2003.
2. M. Litvin and G. Litvin, "*C++ for you*", Vikas Publication, 2002

Digital Logic and Computer Fundamentals

(MAJOR : T)
BCA-201

Paper Title: Digital Logic And Computer Fundamentals
Paper Code: BCA - 201
Number of Hours per Week: (L+T+P = 4+0+0)
Total Contact Hours : 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** Provide a comprehensive understanding of basic concepts and applications in Digital Logic and Computer Fundamentals.
- **CO2:** Cover topics such as Logic Gates, Number Systems, Boolean algebra, Karnaugh Map, Combinational, and Sequential Circuits, Counters, and Registers.
- **CO3:** Equip students with a robust grounding in Digital Logic, fostering the requisite knowledge to engage with advanced concepts in computer architecture and design.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to

- **LO1:** Understand the Fundamentals of Computer Systems.
- **LO2:** Understand the fundamental concepts of Digital Logic Gates.
- **LO3:** Understanding Number Systems and conversion from one number system to other number systems.
- **LO4:** Able to simplify Boolean expressions by means of Algebraic, Karnaugh map and Tabulation methods.
- **LO5:** Understand the basic concept of circuits(Combinational and Sequential).
- **LO6:** Understand the working of Counters and Registers in digital computers.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Introduction to Number System and Digital Logic Gates.	15	18	25
II	Boolean Algebra and Karnaugh Map	15	19	
III	Circuits: Combinational and Sequential	15	19	
IV	Counters and Registers	15	19	
TOTAL		**60**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Number Systems: Bit, Byte, Nibble, Word, Binary Number, Binary Arithmetic (Addition, Subtraction, Multiplication, Division), Subtraction using *r*'s and (*r* - *1*) 's Complement, Hexadecimal number system, Octal number system, Conversion between number systems, Binary codes (BCD, Error-Detection, ASCII, EBCDIC). **Digital Logic Gates**: Logic Gates (AND, OR, NOT, NAND, NOR, XOR and XNOR), Realisation of other logic functions using NAND/NOR gates.

UNIT II **15 Hours**

Boolean Algebra: Boolean variables, postulates and theorems of Boolean Algebra, Boolean functions, Simplification of Boolean expressions by algebraic method, Dual and Complement of a Boolean expression, Canonical form, Standard form, Sum of Products and Product of Sums Forms of Logic expression and conversion between two, conversion of expression in Standard form into Canonical form. **Karnaugh Maps:** Minimization using Karnaugh map for two, three and four variables, Sum-of Products and Products of sums simplification using K-map, Don't Care Conditions, Quine-McCluskey method (Tabulation method).

UNIT III **15 Hours**

Combinational circuits: Introduction, block diagram. Arithmetic Circuits: Half –Adder, Full – Adder. n-to-m line Decoder, Encoder, 2^n-1 Multiplexer, De-multiplexer:2-to-4 line decoder with enable. **Sequential Circuits**: Introduction, block diagram. Flip – Flops: Basic R-S flip flop (Latch), Clocked flip-flops (Logic diagram, Graphic Symbol, Characteristic table, Characteristic equation, Excitation table): R-S flip flop, D flip-flop, J-K flip flop, T flip flop. Master-Slave flip-flop using R-S flip-flop graphic symbols.

UNIT IV **15 Hours**

Counters: Design of a 3-bit binary counter using T flip-flops, 4-bit Binary Ripple counter using J-K flip-flops (Mod-16), BCD Ripple counter. Synchronous Counter: 4-bit binary synchronous counter, 4- bit up-down binary synchronous counter. **Registers**: 4-bit Register, 4-bit Register with parallel load, Shift Register: Serial transfer.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19
IV	2	1	19

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. M. M. Mano, "*Digital Logic and Computer design"*, 5th Edition, Prentice Hall India, 2018.

Reference Books:

1. S. Salivahanan and S. Arivazhagan *"Digital Circuit and Design",* 5th Edition, Oxford University India Press, 2018
2. A. P. Malvino and D.P. Leach, "*Digital Computer and Applications"*, 4th Edition, Tata Mc-Graw Hill Company, 2001.
3. J.P Hayes, *"Computer Architecture and Organisation"*, 4th Edition, Tata Mc-Graw Hill Company, 2001.
4. D. P. Nagpal, *"Computer Fundamentals: Concepts Systems & Applications"*, Wheeler Publishing, 2001.

Semester IV

Data Structure using C
(MAJOR - T + P)
BCA-250

Paper Title: Data Structure using C
Paper Code: BCA-250
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** The course aims to provide a thorough understanding of data structures and their implementation using the C programming language.
- **CO2:** Impart skills to create data structures and optimise data processing for problem-solving.
- **CO3:** Develop proficiency in implementing and manipulating data structures using the C programming language.

Learning Outcomes (LOs):

On successful completion of the course, students will be able to:

- **LO1:** Develop algorithms and analyse time and space complexity to solve a problem.
- **LO2:** Implement various data structures viz. Array, Linked Lists, Stacks, Queues, Trees and Graphs.
- **LO3:** Understanding various Searching and Sorting techniques.
- **LO4:** It is expected that the student has basic knowledge of C Programming.

Outlines of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Design and Analysis of Algorithm; Arrays	15	18	19
II	Linked List, Stacks and Queues	15	19	
III	Trees and Graphs	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Design and Analysis of Algorithm: Concepts of data structure, Abstract Data Type, Algorithms, Different approaches to designing an algorithm, Time and Space Complexity, Big O Notation. **Arrays:** Concept of 1D and 2D Arrays, Representation of 2D Arrays - Row Major and Column Major Representation, Operations on 1D Array - Traversing, Insertion, Deletion, Merging, Searching - Linear Search, Binary Search, Sorting - Bubble Sort, Selection Sort, Insertion Sort, Merge Sort - Divide and Conquer Algorithm.

UNIT II **15 Hours**

Linked List: Concepts of Singly Link List, Singly Linked List Implementations - Insertion, Deletion, Searching, Traversing. Concepts Only - Doubly Linked List, Circular Linked List. **Stacks:** Concepts of Stacks, Stack operations using Array - Push, Pop, Peek, Conversion and Evaluation of Arithmetic Expressions - Polish Notation, Recursion. **Queues:** Concepts of Queues, Queues operations using Array - Insertion and Deletion. Concepts Only - Dequeues, Priority Queues, Circular Queues.

UNIT III **15 Hours**

Trees: Introduction to Trees Concept, Basic Terminology, Binary Trees - Terminology, Complete Binary Trees, 2-Trees, Representation of Binary Trees in the Memory, Traversing A Binary Tree using recursion - Pre-order Traversal, In-order Traversal, Post-order Traversal, Tree Traversal using stacks, Constructing a Binary Tree from Traversal Results. **Binary Search Trees**: Operations on Binary Search Trees - Searching, Insertion, Deletion of Nodes in BST, Concepts Only - Threaded Binary Trees, AVL Trees. **Graphs:** Introduction to Graphs, Graph Terminology, Directed Graphs Terminology, Sequential representation of Graph - Adjacency Matrix, Path matrix, Linked representation of Graphs - Adjacency List, Operations on Graphs - Insertion, Deletion, Searching, Graph Traversal Algorithm - Breadth-first search, Depth-first search. P, NP, NP-Hard and NP-Complete Problems - Basic concepts only

UNIT IV **30 Hours**

Practicals from UNIT I, UNIT II and UNIT III

Suggested Experiments:

1. Write a menu-driven program to:
    a) Read N elements into an array
    b) Print the elements of a given array
    c) Insert an element at a given position
    d) Delete an element from a given position of an array
2. Write a program to merge two unsorted arrays.
3. Write a program, to merge in two sorted arrays.
4. Write a program to search for an element in an array using linear search.
5. Write a program to search for an element in an array using binary search.
6. Write a program to enter ‘n’ numbers in an array. Arrange the numbers in ascending order using the following sorting technique:
    a) Bubble sort
    b) Selection sort
    c) Merge sort
7. Write a program to store the numbers in a 4x4 matrix. Find the sum of the numbers of each row and the sum of the number of each column of the matrix.
8. Write a program to find the product of two matrices.
9. Write a menu-driven program to:
    a) Construct a singly linked list. Assume the information part of each node consists of only an integer key. Get input for each key from the keyboard. Assume the input is over when the user enters –1
    b) Print the information from each node
    c) Delete all nodes containing a given number
    d) Exit
10. Write a program to create a linked list and perform insertion and deletions of all cases.

11. Write a program to search for an element in a linked list. If the number is present then display the message “Search Successful” otherwise “Search is not Successful”
12. Write a C function to insert a node appropriately to an already sorted list so that after insertion, the new list also becomes sorted. Take care of special cases such as inserting into an empty list. Use this function to write a program which accepts integers at the input and at the end produces a sorted list. Assume that if the integer read at the input is ‘0’ then your program should stop.
13. Write a menu-driven program to implement a stack using arrays. The menu should have the following options:
    a) Push on to the stack
    b) Pop from the stack and print the value popped from the stack
    c) Merely print the value on top of the stack
    d) Exit

    Error trapping should be done for underflow and overflow. Available array space should be efficiently used (i.e. there cannot be overflow if there is more than 1 empty element in the array). Assume that the information part of a stack element is only an integer.
14. Write a menu-driven program to implement a stack using arrays. The menu should have the following options:
    a) Push on to the stack
    b) Pop from the stack and print the value popped from the stack
    c) Merely print the value on top of the stack
    d) Exit

    Error trapping should be done for underflow and overflow. Available array space should be efficiently used (i.e. there cannot be overflow if there is more than 1 empty element in the array). Assume that the information part of a stack element is only an integer.
15. Write a program to convert an expression from its infix form to its equivalent
    a) postfix form
    b) prefix form.

    Display the results accordingly.

    Assume the infix expression contains only operators +, -, /, *, ^. The operator ‘^’ stands for exponentiation. The operands are all single digit integers.
16. Write a program to input a postfix expression that consists of only single digit positive operands and the binary operators +, -, *, and /. Using a function, evaluate this postfix expression. The function should report if the postfix expression is invalid, else return its value. [For example, 242/-46*+7+ is a valid postfix expression (being the equivalent of the infix expression, 2-4/2+4*6+7) and its value is 31.00.]
17. Write a program to implement insertion and deletion in a queue
18. Write a program to create a binary tree and to traverse the tree in
    a) pre-order
    b) in-order
    c) post-order
19. Write a program to construct a binary search tree of integers using a linked list. Assume the information part of each node consists of only an integer key. Get input for each key from the keyboard. Assume the input is over when the user enters –1. Next, print out the keys in ascending order of magnitude.
20. Write a program to create a binary search tree and perform the following operations:
    a) Search for a given node.

b) Delete a given node. Display a proper message if a given node is not present in the tree.

21. Write a program to find the biggest and smallest item in a binary search tree.
22. Write a program to create and print a graph using an adjacency matrix.
23. Write a program to create a graph of *n* vertices using an adjacency list and print the value of all the nodes.
24. Write a program to implement the Depth First Search algorithm to traverse a graph.
25. Write a program to implement the Breadth First Search algorithm to traverse a graph.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical:

10% : Syntax and input/output screens
30% : Logic and efficiency (source code, pseudocode and algorithm)
20% : Error Trapping
20% : Completion

20% : Result

Recommended Readings:

Text Books:

1. R. Thareja, "*Data Structure Using C"*, 3rd Edition, Oxford University India Press, 2023
2. S. Lipschutz, "*Data Structures with C"*, 1st Edition, McGraw Hill Education India, 2017

Reference Books:

1. Y. P. Kanetkar, *"Data Structures Through C Language",* 5th Edition, BPB Publications, 2023.
2. Y. Langsam, M. J. Augenstein and A. M. Tenenbaum, *"Data Structures Using C and C++",* 2nd Edition, Prentice Hall of India, 2015.
3. E. Horowitz, S. Sahni and D. Mehta, *"Fundamentals of Data Structures in C",* Universities Press (India) Pvt Ltd, 2009.

Computer System Architecture
(MAJOR - T)
BCA – 251

Paper Title: Computer System Architecture
Paper Code: BCA-251
Number of Hours Per Week: (L+T+P= 4+0+0)
Total Contact Hours: 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** Understand the functional units of a computer, including bus structures, memory operations, and instruction execution in assembly language.
- **CO2:** Explore control memory design, address sequencing, and I/O operations such as programmed I/O, interrupt-initiated I/O, and DMA.
- **CO3:** Learn about pipelining, vector processing, and the concept of parallel processing in different computer models.
- **CO4:** Analyse memory hierarchy, including auxiliary memory, cache memory, virtual memory, and memory management techniques.
- **CO5:** Develop skills in designing and optimising computer systems for efficient data processing, memory utilisation, and I/O operations.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Explain the basics of organisational and architectural issues of a digital computer and Classify and compute the performance of machines, Machine Instructions.
- **LO2:** Describe various data transfer techniques in digital computers and the I/O interfaces.
- **LO3:** Analyse the performance of various classes of Memories, build large memories using small memories for better performance and analyse arithmetic for ALU implementation.
- **LO4:** Describe the basics of hardwired and micro-programmed control of the CPU, and pipelined architectures.

Outline of the Course :

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Basic Concept and Assembly Language	15	18	25
II	Control Unit and Input/Output Organisation.	15	19	
III	Pipeline and Vector Processing	15	19	
IV	Memory Hierarchy	15	19	

Total	60	75	25

CONTENTS

UNIT I **15 Hours**

Basic Concept and Assembly language : Functional units of a computer, Basic operational concepts, Bus structures, memory locations, Address and Encoding of Information, main memory operations, Instructions and Instruction sequencing: Instruction Execution and straight line sequencing, Branching, Addressing modes, Assembly Language: Assembler, Assembler commands, Assembly and execution of programs.

UNIT II **15 Hours**

Control memory, Address sequencing: Routine, mapping, conditioned Branching, mapping instruction, Design of control unit: Microprogram sequencer. **Input Output**: Accessing I/O devices, Data transfer between the CPU and I/O devices: Programmed I/O, Interrupt initiated I/O and Direct Memory Access (DMA). Memory mapped I/O vs. Isolated I/O.

UNIT III **15 Hours**

Pipelining and vector processing: Four models of Computers (SISD, SIMD, MISD, MIMD), Parallel processing, pipeline. Arithmetic pipeline, Instruction pipeline, RISC pipeline, Vector processing; vector operations, Memory Interleaving, concept of Supercomputers.

UNIT IV **15 Hours**

Memory Hierarchy: auxiliary memory, cache memory, multiprogramming; main memory, RAM, ROM, Bootstrap loader, computer start up. RAM and ROM chips, memory Address map, memory connection to CPU; auxiliary memory; magnetic disk, magnetic tape; Associative memory; cache memory; Locality of reference, hit ratio, mapping, Associative mapping, Direct mapping, writing into cache; virtual memory; Address space and memory space, Address mapping using papers, Associative memory page table; memory management hardware, memory protection

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19
IV	2	1	19

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. M. M. Mano, *"Computer System Architecture"*, 3rd Edition, Pearson, 2017.
2. C. Hamacher, Z. Vranesic and S. Zaky, *"Computer Architecture And Organization"*, 5th Edition, McGraw Hill, New Delhi, India, 2002.

Reference Books:

1. H. Stone, *"Introduction to Computer architecture"*, 3rd Edition, Galgotia Publishing Ltd., 2001
2. P. P. Chaudhuri, *"Computer Organization and Design"*, 4th Edition, Prentice Hall India, 2002.
3. T. C. Bartee, *"Computer Architecture and Logic Design"*, Tata McGraw-Hill, International Edition, 2001.
4. B. Ram, *"Computer Fundamentals, Architecture and Organization"*, 3rd Edition, New Age International Publishers, 2002.

Database Management Systems
(MAJOT - T + P)
BCA-252

Paper Title : Database Management Systems
Paper Code : BCA-252
Number of Hours Per Week : (L+T+P = 3+0+2)
Number of contact hours : 75
Number of Credits : 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Understand the fundamental concepts of Database Management Systems (DBMS) and database design principles using Entity-Relationship (ER) diagrams
- **CO2:** Explore file organisation techniques, Relational Model, relational algebra operations and normalisation techniques up to BCNF.
- **CO3:** Acquire knowledge of transaction processing concepts, concurrency control techniques, schedules, recoverability, and deadlock prevention mechanisms in database systems.

Learning Outcomes (LOs):

On successful completion of the course, students will be able to:

- **LO1:** Understand the fundamental concepts of relational database management systems (RDBMS) and their applications.
- **LO2:** Demonstrate proficiency in designing entity-relationship (ER) models to represent real-world scenarios.
- **LO3:** Utilise SQL to manipulate and query relational databases effectively.
- **LO4:** Identify functional dependencies and apply normalisation techniques to achieve higher database integrity.
- **LO4:** Develop practical skills through hands-on exercises and projects using a popular RDBMS software.

Outlines of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Introduction to DBMS Concepts, Database Design and File Organization	15	18	19
II	Relational Algebra Concepts, Functional Dependency and Normal Forms	15	19	
III	Transaction Processing Concepts, Schedules and Recoverability, Concurrency Control Techniques	15	19	
IV	Practical	30	19	6
TOTAL		**75**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Introduction to DBMS Concepts: File Systems versus DBMS, the Data Model, Levels of Abstraction, Data Independence, Structure of a DBMS. **Introduction to Database Design:** Database Design and E-R Diagrams, Entities, Attributes, Entity Types, Value sets, Key attributes, Relationships, Relationship Sets, Additional Features of the ER Model, Conceptual Design with ER Model. **File Organization and Indexing:** Types of single-level Ordered Indexes: Primary Indexes, Clustering Indexes; Multilevel Indexes, search trees and B-trees.

UNIT II **15 hours**

Relational Model Concepts: Domains, Tuples, Attributes, and Relations, Characteristics of Relations, Relational Model Notation; Relational Model constraints, update operations on relations; Insert, Delete, Modify operations; Defining Relations, **Relational Algebra**: SELECT, PROJECT, UNION, INTERSECTION, DIFFERENCE, JOIN, DIVISION operations. **Functional dependency and Normal forms**: Definition of Functional dependency, Interference Rules for Functional Dependencies, equivalence of sets of Functional Dependencies, Minimal sets of Functional Dependencies, Introduction to Normalisation, Definitions of INF, 2NF, 3NF, BCNF.

UNIT III **15 hours**

Transaction Processing Concepts: Introduction to Transaction Processing, Read & Write Operations, Need for Concurrency Control, Transaction States, Commit Point of a Transaction, Desirable Transaction Properties. **Schedules and Recoverability**: Schedules, Characterising Schedules, Serializability of Schedules, Testing for Conflict Serializability of a Schedule, Uses of Serializability, **Concurrency Control Techniques:** The Locking Protocol, Locks. Two Phase Locking (2PL), Deadlock and its Prevention

UNIT IV **30 hours**

Suggested Experiments:

1. A Case Study with Report to be presented by individual student covering the following concepts
    a. ER Diagram
    b. Relational Model
    c. Normalisation

2. Practical Assignments: Use any appropriate RDBMS to be made available – preferably MySQL)
    a. Exercises using DDL Commands
    b. Exercises using DML Commands
    c. Querying using ANY, ALL, IN, Exists, Not Exists, Union, Intersection Constraints etc.
    d. Querying using Aggregate functions, group by, having and Creation and dropping of views.

e. Triggers (Creation of insert trigger, delete trigger, update trigger)
f. Procedures

Distribution of marks for practical:

10% : Syntax and input/output screens
30% : Logic and efficiency (source code, pseudocode and algorithm)
20% : Error Trapping
20% : Completion
20% : Result

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

Topic	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
Topic: Creating databases, tables and insertion of data	1	1	6
Topic: SQL queries and views related to Q1.	2	1	6
Topic: Triggers and Procedures related to Q1.	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Recommended Reading:

Text Books:

1. R. Elmasri and B. Navathe, *"Fundamentals of Database System"*, 7th Edition, Addition-Wesley, 2017
2. A. Silberschatz and H. F. Korth, *"Database System Concepts"*, 5th edition, McGraw Hill, 2021

References Books:

1. R. Ramkrishnan and J. Gehrke, *"Database Management Systems"*, 3rd Edition, Tata Mc Graw Hill, 2002.
2. C.J. Date, A. Kannan and S. Swamynathan, *"Introduction to Database Systems"*, 8th Edition, Pearson Education, 2006.
3. M. Gillenson, *"Fundamentals of Database Management Systems"*, 2nd Edition, Wiley Student Edition, 2012

Operating Systems and Shell Programming
(MAJOR - T + P)
BCA - 253

Paper Title: Operating Systems and Shell Programming.
Paper Code: BCA - 253
Number of Hours per Week: (L+T+P = 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Develop an understanding of operating systems, covering various system types, structures, process management, synchronisation, CPU scheduling, deadlock resolution, threads, memory management, paging, demand paging, and file systems.
- **CO2:** Understand the concepts such as processes, inter-process communication, synchronisation techniques, CPU scheduling algorithms, deadlock handling strategies, memory management techniques including paging and demand paging, and file system operations.
- **CO3:** Learning basic linux commands and shell programming.

Learning Outcomes (LOs):

On successful completion of the course, students will be able to

- **LO1:** Understand the importance of computer system resources and the role of operating systems in their management policies and algorithms.
- **LO2:** Understand various process management concepts including scheduling, synchronisation, and deadlocks.
- **LO3:** Understand memory management concepts.
- **LO4:** Understand the file systems, access methods, directory structures, and file system implementation techniques, which are essential for organising and managing data on storage devices.
- **LO5:** Develop shell scripting skills through hands-on exercise using vim editor that enhances problem-solving abilities.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Overview of Operating Systems	15	18	19
II	Process Management, Process Synchronisation, CPU Scheduling, Deadlocks, Threads	15	19	
III	Memory Management, File System,	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Overview of Operating Systems: Simple Batch Systems, Time Sharing Systems, Personal Computer Systems, Parallel Systems, Distributed Systems, Real Time Systems. Operating System Structures: System Components and services, system calls, system programs, virtual Machines.

UNIT II **15 hours**

Process Management: Process Concept, Operation on Processes, Cooperating Processes, Inter-process Communication. **Process Synchronisation:** The critical section problem, two process solution and multiple process solution. Synchronisation hardware, mutex locks, Semaphores—implementation, some critical problems of synchronisation. Readers and Writers problem, Dining-Philosophers' problem. **CPU Scheduling:** Basic Concepts, scheduling criteria, scheduling algorithms, FCFS, SJF, RR, Priority Scheduling. **Deadlocks:** Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock. **Threads:** Overview and benefits.

UNIT III **15 hours**

Memory Management: Logical and physical address space, Swapping, Contiguous allocation – memory protection, memory allocation, fragmentation. Paging: Basic method, hardware support, protection. Demand Paging: Basic concepts, basic scheme of page replacement, page replacement algorithms. **File System:** File Concept (File Attributes, File Operations, File Types), Directory and Disk Structure (Basic Concepts), File-System Mounting, File Protection (Types of Access, Access Control).

UNIT IV **30 hours**

Basic shell Commands in Linux:

- Elementary Linux Utilities: cal, date, who, uname, passwd, echo, tput, bc.
- Elementary Linux commands; cd, mkdir, pwd, rmdir, chmod, chown, ls, cat, cp, rm, more, wc, split, cmp.
- Simple Filters: pr, head, tail, cut, paste, sort, uniq, tr.

- **Shell Scripting: Assignments covering the following concepts to be completed**
- Introduction to shell script: Shell variables, data types
- String and array manipulation, Control structures, Functions, Command -Line Arguments and Options, Input and Output Redirection, Pipes, Tees, Command Substitution.
- Introduction to regular expressions
- Pattern matching and substitution: use of the following tools: sed. awk, grep
- Error handling and debugging techniques

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of questions to be set from Shell Scripting according to the following scheme

Shell Scripting Questions from UNIT-IV	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
Q1	2	1	6
Q2	2	1	6
Q3	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical:

10% : Syntax and input/output screens
30% : Logic and efficiency (source code, pseudocode and algorithm)
20% : Error Trapping
20% : Completion
20% : Result

Recommended Reading:

Text Books:

1. A. Silberschatz, P. B. Galvin and G. Gagne, *"Operating System Concepts"*, 9th Edition, Addison-Wesley, 2018.
2. S. Das, *"Unix: Concepts and Applications"*, 4th Edition, Tata McGraw Hill, 2006

Reference Books:

1. A. Tanenbaum, *"Modern Operating Systems"*, 4th Edition, Pearson Education Inc., 2009.
2. W. Stallings, *"Operating Systems"*, 7th Edition, Prentice Hall India Pvt Ltd, 2012

Semester V

Software Engineering
(MAJOR - T)
BCA-300

Paper Title: Software Engineering
Paper Code: BCA-300
Number of Hours Per Week: (L+T+P= 4+0+0)
Total Contact Hours: 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** Understand system concepts, types, and the role of a systems analyst, including data gathering techniques and software life cycle models.
- **CO2:** Gain knowledge of software project management principles, including project planning, estimation techniques, risk management, configuration management, and project scheduling.
- **CO3:** Learn about requirement analysis and specification, function-oriented and object-oriented software design methodologies, user interface design principles, and GUI development.
- **CO4:** Develop skills in coding, testing, software documentation, software reliability, and software maintenance, including testing techniques, debugging, integration testing, and software quality management.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Understand the principles and practices of software engineering.
- **LO2:** Apply software engineering processes and methodologies to develop software systems.
- **LO3:** Perform requirements analysis and software design.
- **LO4:** Implement software using appropriate programming languages and development tools.
- **LO5:** Apply software testing and quality assurance techniques.

Outline of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Introduction, System Analysis and Design	15	18	25
II	Project Management	15	19	
III	Function and Interface Design	15	19	
IV	Coding,Testing and Maintenance	15	19	
Total		**60**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Introduction: System definition and concepts: Characteristics and types of system, Manual and automated, MIS, DSS, Real Time and Distributed. **Systems analyst**: Roles and Responsibilities of Systems Analyst , Data and fact gathering techniques used by Systems Analyst. **Software Life cycle models**: Roles and Responsibilities of Systems Analyst, Data and fact gathering techniques used by Systems Analyst. **Software Life cycle models**: Importance of a life cycle model, waterfall model (feasibility study, requirement , analysis and specification, design, coding and unit testing, integration and system testing, maintenance), prototyping model, evolutionary model, spiral model, Comparison of different life cycle models.

UNIT II **15 hours**

Software Project Management: Responsibilities of a Software Project Manager, Project Planning, Project Estimation Techniques Size Estimation like lines of Code & Function Count, Cost Estimation Models, COCOMO, Risk Management, Software Configuration Management. **Project Scheduling:** Work breakdown, Activity Networks and Critical Path Method, Gantt Charts, PERT Charts, Project Monitoring and Control. **Requirement Analysis and specification**: Software Requirements Specification (Content of the SRS document, characteristics of a good SRS document, techniques for representing complex logic – Decision Tree and Decision Table).

UNIT III **15 hours**

Function Oriented Software Design: Overview of SA/SD methodology, Structured Analysis, data Flow Diagrams (DFDs)(primitive symbols used for constructing DFDs, important concepts associated with designing DFDs, developing the DFD Model of a system, Shortcomings of the DFD Model). Object Oriented Design. **User Interface Design**: characteristics of a good user interface, basic concepts (user guidance and online help, mode-based vs. Modeless Interface, Graphical User Interface (GUI) vs. Text-based User Interface), Types of user interfaces (command language-based Interface, Menu-based Interface, direct manipulation Interface), Component-Based GUI Development (Window system, Types of widgets, Visual programming), User interface methodology (Design, a GUI design methodology).

UNIT IV **15 hours**

Coding and Testing: coding standards and guidelines, code review (code walkthroughs, code inspection), Software Documentation(Internal and External), Testing (testing, verification vs. validation, design of test cases), Testing in the large, Testing in the small, unit testing, Black-box testing, White-box testing (Statement, Branch and Condition coverage), debugging, integration testing, system testing. **Software Reliability**: Software reliability, software quality, and software quality management. **Software Maintenance**: Characteristics of software maintenance (types of software maintenance, special problems associated with software maintenance), software reverse engineering.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	18
II	2	1	19
III	2	1	19
IV	2	1	19

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. R. Pressman, *"Software Engineering: A Practitioner's Approach",* 9th Edition, McGraw Hill, 2023
2. R. Mall, *"Fundamentals of Software Engineering"*, 5th Edition, Pearson Education / Prentice Hall of India, New Delhi, 2018.

References Books:

1. E. M. Awad, *"System Analysis and Design"*, 2nd Edition, Galgotia Publications (P) Ltd, New Delhi, 2008.
2. C. Ghezzi, M. Jazayeri and D. Mandrioli, *"Fundamentals of Software Engineering"*, 2nd Edition, Prentice Hall of India Private Limited, New Delhi, 2002.
3. R. E. Fairley, *"Software Engineering Concepts"*, Tata McGraw Hill Publishing Company Limited, New Delhi, 1997.

Mathematics I
(MAJOR - T+P)
BCA-301

Paper Title: Mathematics I
Paper Code: BCA-301
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Develop a foundational understanding of fundamental mathematical concepts including set theory, mathematical logic, counting principles, statistics, and probability.
- **CO2:** Apply mathematical reasoning and problem-solving skills to analyse and solve problems across various domains, particularly in business, industry, and scientific research.
- **CO3:** Equip students with the necessary tools and techniques to interpret and communicate mathematical information effectively, both orally and in written form, and to make informed decisions based on statistical and probabilistic analyses.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Understand fundamental concepts of set theory, relations, and functions, including operations, properties, and applications.
- **LO2:** Apply principles of mathematical logic to analyse propositions, construct logical arguments, and perform proofs.
- **LO3:** Utilise counting principles and techniques to solve problems involving permutations, combinations, and the Pigeonhole Principle.
- **LO4:** Analyse statistical data using graphical representations, compute measures of central tendency and dispersion, and interpret their significance in practical contexts.
- **LO5:** Apply probability concepts to compute probabilities, joint distributions, and moments, and understand the applications of special probability distributions in various scenarios.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Set Theory, Functions and Graphs	15	18	19
II	Mathematical Logic	15	19	
III	Statistics and Probability	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT-I **15 Hours**

Set Theory: Basics in set theory such as ways of describing a set, set operations, empty set, disjoint sets, De Morgan's laws,Venn diagrams; power sets, Cartesian products, cardinality results; relation as a subset of Cartesian product (notation: xRy if (x,y) Є R); relation on a set: reflexive, symmetric, anti-symmetric, transitive with examples. Equivalence relation. **Functions and Graphs:** Real valued functions such as polynomials, rational functions, logarithmic functions,exponential functions, limits, standard theorems on limits, standard limits. Continuity and its properties, differentiability. Statements with applications only of the following: Boundedness, Intermediate value theorem.

UNIT-II **15 Hours**

Mathematical Logic: Connectives, well formed formulas, truth tables, tautology, equivalence, implication, normal forms, predicates, free & bound variables, rules of inference, consistency, Proof by contradiction. Counting and Relations: Basics of counting, Pigeonhole Principle, Permutation and Combinations, Binomial coefficients

UNIT-III **15 Hours**

Statistics: Definition of Statistics, Nature and scope of statistics, uses of statistics in business and industrial activities, Statistical Data, collection of statistical data, representation of statistical data-bar charts, pie diagram, line graphTabulation of data frequency distribution table. Graphical representation of frequency distribution-histogram, frequency polygon, cumulative frequency distribution and the ogive. Measures of central tendency-mean, median, mode and their applications in business. Measures of dispersion-range, quartile deviation, mean deviation, standard deviation, coefficient of variation, uses of dispersion. **Probability:** Probability elementary concepts of probability, probability density functions,distributionfunction,jointdistribution,moment generating and characteristic functions, Mathematical expectation,special probability distribution, Normal, Exponential, Hypergeometric distributions.

UNIT-IV **30 Hours**

(Any appropriate statistical software package such as R may be used)

Suggested Experiments:

Data Collection and Representation:

1. Utilise statistical software to analyse and visualise datasets related to business or industrial activities.
1. Generate bar charts, pie diagrams, and line graphs to represent the collected data effectively.
2. Compare and contrast different graphical representations produced by the software to determine the most suitable for conveying specific types of information.

Frequency Distribution and Graphical Representation:

3. Use statistical software to create frequency distribution tables from given datasets.

4. Generate histograms, frequency polygons, and cumulative frequency distributions based on the frequency distribution tables.
5. Interpret the graphical representations generated by the software to identify patterns and trends in the data.

Measures of Central Tendency and Dispersion:

6. Utilise statistical software to calculate the mean, median, and mode for various datasets representing business or industrial scenarios.
7. Discuss the practical applications of each measure of central tendency in analysing sales data or employee performance, supported by software-generated results.
8. Compute measures of dispersion such as range, quartile deviation, mean deviation, standard deviation, and coefficient of variation using statistical software and interpret their implications for variability and spread within the data.

Probability Concepts and Distributions:

9. Explore elementary probability concepts using statistical software through simulated experiments or random sampling.
10. Calculate probabilities for various events using built-in probability density functions and distribution functions.
11. Utilise statistical software to compute joint distributions and moments for pairs of random variables in relevant business or industrial contexts.
12. Apply the concepts of mathematical expectation using statistical software to calculate expected values in scenarios such as project management or inventory control.
13. Analyse the characteristics and applications of special probability distributions like the Normal, Exponential, and Hypergeometric distributions using statistical software, focusing on real-world scenarios such as modelling product lifetimes or analysing defect rates in manufacturing processes.
14. Generate and plot probability density functions (PDFs) for continuous random variables such as uniform, normal, and exponential distributions.
15. Generate random samples from special distributions such as the normal, exponential, and hypergeometric distributions using software tools.
16. Analyse the properties of MGFs and their relationship with moments of random variables.

Integration of Statistics and Probability:

17. Employ statistical software to design and conduct experiments, and gather data relevant to business or industrial activities.
18. Apply statistical analysis techniques, including measures of central tendency, dispersion, and probability distributions within the software environment, to draw conclusions and make predictions based on the collected data.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**MARKS**
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

Questions to be prepared from a list of practical given in UNIT-IV	**QUESTIONS**		
	TO BE SET	**TO BE ANSWERED**	**MARKS**
Q1. Topic: Data Collection and Representation, Frequency Distribution and Graphical Representation	2	1	6
Q2. Measures of Central Tendency and Dispersion	2	1	6
Q3. Probability Concepts and Distributions, Integration of Statistics and Probability	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. J. P. Trembly and R.P. Manohar, *"Discrete Mathematical Structures with Applications to Computer Science"*, International Edition, McGraw-Hill, 2001.
2. E. Kreyszig, *"Advanced Engineering Mathematics"*, 6th Edition, Wiley International, 2002.
3. M. K. Bhowal and P Barua, *"A First Course in Statistics Volume 1"*, 2nd Edition, 2019.

Reference Books:

1. B. K. Pal and K. Das, *"BCA Mathematics Vol. IV"*, 1st Edition, U.N. Dhur & Sons Private Ltd., 2011.

Advanced Python Programming
(MAJOR / MINOR - T + P)
BCA-302

Paper Title: Advanced Python Programming
Paper Code: BCA-302
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3+ Pr: 1)

Course Objectives (COs):

- **CO1:** This course is designed to introduce programming concepts using Python. The course focuses on the development of Python programming to solve problems of different domains.
- **CO2:** Gain robust understanding of the concept of object-oriented concepts as applied in python programming
- **CO3:** Develop proficiency in using python libraries such as Numpy, Pandas and Matplotlib

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Explain the basic concepts of Python Programming.
- **LO2:** Demonstrate proficiency in the handling of loops and creation of functions.
- **LO3:** Identify the methods to create and manipulate lists, tuples and dictionaries.
- **LO4:** Apply Numpy, Pandas and Matplotlib
- **LO5:** Interpret the concepts of Object-Oriented Programming as used in Python.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Basics of Python, Control Flow, Loops Strings and Functions	15	18	19
II	List, Dictionaries, Tuples, Sets, Numpy, Pandas and Matplotlib	15	19	
III	OOP Concepts and Exception	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Python Basics: Identifiers, Keywords, Statements and Expressions, Variables, Operators, Precedence and Association, Data Types, Indentation, Comments, Built-in Functions-

Console Input and Console Output, Type Conversions, Python Libraries, Importing Libraries with Examples. **Python Control Flow**: Types of Control Flow; Control Flow Statements- if, else, elif, while loop, break, continue statements, for loop Statement; **Strings:** Creating and Storing Strings; Accessing String Characters; the str() function; Operations on Strings- Concatenation, Comparison, Slicing and Joining, Traversing, Format Specifiers, Escape Sequences, Python String Methods. **Python Functions:** Types of Functions; Function Definition- Syntax, Function Calling, Passing Parameters/arguments, the return statement; Default Parameters; Recursive Functions

UNIT II **15 hours**

Lists: Creating Lists; Nested Lists, Operations on Lists, Basic Built-in Functions on Lists. **Dictionaries**: Creating Dictionaries; Operations on Dictionaries; Basic Built-in Functions on Dictionaries, Populating and Traversing Dictionaries. **Tuplesand Sets:** Creating Tuples; Operations on Tuples; Built-in Functions on Tuples; Tuple Methods; Creating Sets; Operations on Sets; Basic Built-in Functions on Sets. **NumPy**: Introduction to NumPy, Array Creation using NumPy, Operations on Arrays. **Pandas**: Introduction to Pandas, Series and DataFrames, Creating DataFrames from Excel Sheet and .csv file, Dictionary and Tuples. Basic Operations on DataFrames. **Data Visualisation:** Introduction to Data Visualisation, Matplotlib Library; Different Types of Charts using Pyplot- Line chart, Bar chart and Histogram and Pie chart

UNIT III **15 hours**

OOP Concepts: Overview of OOP concepts, Defining Classes, Creating Objects, Constructor (_init_), class methods, class variables and object variables, Access modifiers – public, private, and protected, getter and setter methods, Static methods, Inheritance, types of inheritance- single inheritance, multiple inheritance, multi-level inheritance, multipath inheritance, polymorphism and method overloading, super keyword. **Exceptions:** Exceptions, Handling exceptions, multiple except blocks, raising exceptions, built-in exceptions and user-defined exceptions

UNIT IV **30 hours**

Practical from UNIT I, UNIT II and UNIT III.

Suggested Basic Experiments:

1. Create a program that takes user input for the radius of a circle and calculates its area using variables and basic arithmetic operators.
2. Create a program to display the first n Fibonacci number.
3. Write a program to check if a number is an Armstrong number.
4. Write a program that checks whether a given number is positive, negative, or zero using if-else statements.
5. Write a Python program that takes a student's score as input and prints their corresponding grade based on the following criteria:
   A: 90-100
   B: 80-89
   C: 70-79
   D: 60-69
   F: Below 60
6. Develop a program that takes a user-entered sentence and performs operations like string concatenation, slicing, and checks for the presence of a specific word.
7. Create a function to calculate the factorial of a given number using a recursive

function.

8. Design a program that accepts user input as a string and converts it into different data types (int, float) using type conversion functions.
9. Implement a program that checks if a number is prime using a while loop for repeated checks and breaks out of the loop when a prime number is found.
10. Write a program to print a pyramid design with *.
11. Write a program to check if a word is a palindrome.
12. Write a program that takes a user's full name as input, capitalises the first letter of each word using string methods, and displays the formatted name.
13. Write a program to count the frequency of each word in a sentence and remove duplicate words
14. Write programs to illustrate the use of function calls with arguments and functions returning values.
15. Write a program to take input for a sentence and sort the words according to their length.

Suggested Advanced Experiments:

1. Write a program to perform matrix addition and multiplication using nested lists.
2. Implement a simple shopping cart system using lists to add, remove, and display items.
3. Create a program that takes multiple lists as input and calculates the length of each list using a function.
4. Write a program to build a contact book program using dictionaries to store and manage contact details. Allow users to update phone numbers in the contact book program.
5. Write a function to print all keys and values from a dictionary in separate lists.
6. Write a program that takes user input for a day and checks if it's a weekday using a tuple.
7. Implement a program that performs set operations (union, intersection, difference) on two sets.
8. Use NumPy to calculate the mean, median, and standard deviation of a dataset.
9. Write a program that reshapes a 1D NumPy array into a 2D array.
10. Load a dataset into a Pandas DataFrame and explore basic statistics and information about the data. Add a new column to a DataFrame based on a calculation using existing columns.
11. Use Matplotlib to plot the trend of a dataset over time (line chart), create a bar chart comparing the sales of different products, plot a histogram to analyse the distribution of ages in a dataset and display the percentage distribution of different categories using a pie chart.
12. Define a Python class named Car with attributes such as make, model, and year. Include a method within the class to display the car's information. Create an object of the Car class and invoke the method to print the details.
13. Create a base class called Animal with a method make_sound. Create two derived classes, Dog and Cat, that inherit from the Animal class. Override the make_sound method in both derived classes to represent the sounds a dog and a cat make. Instantiate objects of both derived classes and invoke the make_sound method.
14. Define a class called Person with private attributes (_name and _age). Implement a property setter for the _age attribute to ensure that the age is a positive integer. Create an object of the Person class, set the name and age using the property setter, and then print the person's details.

15. Write a program that takes two numbers as input from the user and performs division. Handle the following exceptions: ZeroDivisionError: If the user attempts to divide by zero, print an error message indicating that division by zero is not allowed. ValueError: If the user enters a non-numeric value, print an error message indicating that only numeric values are allowed. Exception: If any other unexpected exception occurs, print a general error message.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. R. Thareja, *"Python Programming: Using Problem-Solving Approach"*, 2nd Edition, Oxford University Press India, 2023
2. S. Gowrishankar and A. Veena, *"Introduction to Python Programming"*, CRC Press, 2019.
3. F. Nelli, "*Python Data Analytics: Data Analysis and Science Using Pandas, matplotlib, and the Python Programming Language",* 1st Edition, Apress, 2015

Reference Books:

1. T. A. Budd, *"Exploring Python"*, 1st Edition, McGraw Hill, 2011
2. M. Lutz, *"Learning Python: Powerful Object-Oriented Programming"*, 5th Edition, Shroff/O'Reilly, 2013

Semester VI

Mathematics II
(MAJOR - T)
BCA-350

Paper Title : Mathematics II
Paper Code: BCA-350
Number of Hours Per Week: (L+T+P= 4+0+0)
Total Contact Hours: 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** Learn to manipulate vectors in 2D and 3D spaces, including addition, scalar multiplication, dot product, and cross product, and apply them to solve problems in geometry and trigonometry.
- **CO2:** Understand the properties and operations of matrices, including addition, multiplication, transpose, and special types of matrices. Learn to calculate determinants, inverses, rank, and eigenvalues of matrices.
- **CO3:** Study differential calculus to find derivatives of functions and solve problems related to maxima, minima, tangents, and normals. Learn integral calculus to find antiderivatives, properties of definite integrals, and improper integrals.

Learning Outcomes (LOs):

- **LO1:** On successful completion of the course, students will be able to:
- **LO2:** Master operations and applications of vectors in various coordinate systems for geometry and physics.
- **LO3:** Understand matrix operations, determinants, and inverses for solving linear equations and transformations.
- **LO4:** Apply differential calculus for derivatives and optimization, and understand basic integral calculus for problem-solving.
- **LO5:** Develop strong problem-solving abilities and critical thinking skills through practical applications.
- **LO6:** Use mathematical models to analyse real-world problems, preparing for complex engineering and scientific challenges.

Outlines of the Course:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Vectors, Coordinate Geometry	15	19	25
II	Matrices	15	18	
III	Differential Calculus	15	19	
IV	Integral Calculus	15	19	
Total		**60**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Vectors: Space Coordinate: rectangular, cartesian, cylindrical, spherical, polar. Vectors (in space) as directed line segments drawn from a fixed point, addition and scalar multiplication. Free vectors and localised vectors, Section formula for position vectors. Dot/Scalar product. cross product. Projection of a line segment. **2-D Geometry**: Basic concepts of straight line, circle, ellipse, parabola and hyperbola. **3-D Geometry:** Basic concepts of plane.

UNIT II **15 hours**

Matrices: Definitions, Addition, multiplication, transpose, conjugate transpose; special type of matrices: diagonal, scalar, upper/lower triangular, nilpotent, idempotent, symmetric, skew symmetric, hermitian, skew hermitian matrices; trace of a square matrix; Adjoint, Singular and Non singular matrix, Inverse of a matrix Orthogonal matrix, Elementary transformation, Rank, Normal form, Cramer's Rule, Eigen Vectors of a Matrix. Linear dependence and independence of vectors, Subspaces and bases and dimensions, Orthogonal bases and orthogonal projections , Gram-Schmidt process.

UNIT III **15 hours**

Differential Calculus: Derivatives of real valued functions on intervals: definition; derivative as a rate measurer, derivative as the gradient of tangent. Working out the derivatives of the common functions mentioned in UNIT-1; Review of methods of differentiation. Differentials, maxima and minima of functions,tangents and normal, L'Hospital's Rule (statements only with applications).

UNIT IV **15 hours**

Integral Calculus: Antiderivatives - Review of the standard methods, integration by parts and by partial fractions. Integral of a continuous function as the limit of sum (only for sums arising out of equal distribution of intervals); examples of evaluation of integrals from the definition. Statements with illustrations of the properties of definite integral, evaluation of integrals using these properties. Improper integrals, convergence and evaluation from definition.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19

III	2	1	19
IV	2	1	19

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. A. R. Vasishtha and A. K. Vasishtha, *"Matrices"*, New Edition, Krishna's Educational Publishers, 2013
2. B. Das, *"Analytical Geometry and vector Analysis"*, 4th Edition, Orient Book Co., 2001
3. K. C. Maity and R.K. Ghosh, *"Integral Calculus"*,6th Edition, New Central Book Agency(P) Ltd., 2003
4. K. C. Maity and R.K. Ghosh, *"Differential Calculus"*, 6th Edition, New Central Book Agency(P) Ltd., 2004

Reference Books:

1. B. K. Pal and K. Das, *"BCA Mathematics Vol. IV"*, 1st Edition, U.N. Dhur & Sons Private Ltd., 2011
2. J. R. F. Ayres, *"Matrices"*, Schaum's Outline Series, 5th Edition, 2002

Data Communications and Networking
(MAJOR - T)
BCA - 351

Paper Title: Data Communications and Networking
Paper Code: BCA-351
Number of Hours Per Week: (L+T+P= 4+0+0)
Total Contact Hours: 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** Understand the fundamentals of data communications, networks, protocols, and standards, including network models like the OSI Model and TCP/IP Protocol Suite, addressing schemes, and physical media types.
- **CO2:** Explore the Data Link Layer concepts such as error detection and correction, framing methods, flow control protocols, multiple access protocols, and channelization techniques.
- **CO3:** Gain knowledge of the Network Layer, including logical addressing, internet protocols, delivery, forwarding, routing techniques, the Transport Layer with process-to-process delivery, and various application layer protocols like DNS, TELNET, FTP, and HTTP.

Learning Outcomes (LOs):

Upon successful completion of the course, students will be able to understand the following concepts:

- **LO1:** Network communication using the layered model concept.
- Various types of transmission media and network devices.
- **LO2:** Flow control, error control, and routing methods.
- **LO3:** Principles and operations behind various application layer protocols such as HTTP, DNS, SMTP, and FTP.

Outline of the Course:

UNIT	Topic	Hours	Ext. Marks	Int. Marks
I	Introduction, Physical layer and Media	15	18	25
II	Data link layer	15	19	
III	Network layer	15	19	
IV	Transport and application layer	15	19	
Total		**60**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Introduction: Data Communications, Networks, Protocols and Standards, Network Models: Layered Tasks, OSI Model, Layers in the OSI Model, TCP/IP Protocol Suite, Addressing. **Physical Layer and Media:** Analog and Digital Signals, Data Rate Limits: Nyquist and Shannon's law, Multiplexing: FDM, TDM, Guided and Unguided media: Twisted Pair, Coaxial, Fibre Optics, Radio, microwave, infrared, Network connecting devices hub,

repeater, bridge, switch, router, and gateway, Switching: Circuit Switched Networks, Datagram Networks, Virtual Circuit Networks.

UNIT II **15 Hours**

Data Link Layer: Error Detection and Correction: Error Correction using Hamming code, Error detecting using CRC, Framing and Framing Methods, Concept of Flow Control, Simplex protocol, Stop-and-Wait Protocol, Stop and Wait ARQ Protocol, Go back N ARQ Protocol, Selective Repeat Protocol, Piggybacking. **Multiple Access Protocols:** Random Access: Pure ALOHA, Slotted ALOHA, Carrier Sense Protocols (I-persistent, p-persistent, and Non-Persistent CSMA), CSMA/CD, CSMA/CA. Controlled Access: Reservation, Polling, Token Passing, Channelization: FDMA, TDMA.

UNIT III **15 Hours**

Network Layer: Logical Addressing: IPv4 Addresses, Address Mapping: ARP, RARP, BOOTP, DHCP, ICMP, IGMP. **Internet protocol:** Internetworking, IPv4. Delivery, Forwarding, and Routing: Delivery, Forwarding, Unicast Routing Protocol: Distance Vector Routing, the Count-to-Infinity problem, Link State Routing, Path Vector Routing.

UNIT IV **15 Hours**

Transport Layer: Process to Process Delivery: Client/Server Paradigm, Multiplexing and Demultiplexing, Connectionless Vs Connection Oriented Service, UDP: User Datagram, Checksum, UDP operation, use of UDP; TCP: Services, features, segment, TCP connection, Flow control, Error control. **Application Layer:** DNS: Name Space, Domain Name Space, Distribution of name space, DNS on the internet, Resolution, Types of Record. TELNET, Electronic Mail, FTP, HTTP.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19
IV	2	1	19

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. B. A. Forouzan, *"Data Communications and Networking"*, Tata McGraw Hill, 6th Edition, Special Indian Edition, 2022.
2. A. S. Tanenbaum, N. Feamster and D. J. Wetherall *"Computer Networks"*, 6th Edition, Pearson, 2022

Reference Books:

1. B. Trivedi, *"Data Communication and Networks"*, 1st Edition, Oxford University Press, 2016
2. W. Stallings, *"Data and Computer Communications"*, 10th Edition, Pearson Education, 2017

Object Oriented Programming in Java
(MAJOR - T + P)
BCA - 352

Paper Title: Object Oriented Programming in Java
Paper Code: BCA-352
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** To understand object-oriented programming concepts and apply them in solving problems.
- **CO2:** To introduce the principles of object-oriented programming; and demonstrate their implementation through classes, inheritance, polymorphism.
- **CO3:** To introduce the implementation of packages and interfaces
- **CO4:** To introduce the concepts of exception handling, multithreading, socket programming.
- **CO5:** To introduce working with Generic classes and methods.
- **CO6:** To introduce the working of databases using Java.

Learning Outcomes (LOs):

- **LO1:** Solving real world problems using OOP techniques.
- **LO2:** Able to handle and understand JAVA exceptions.
- **LO3:** Able to develop multithreaded applications with synchronisation.
- **LO4:** Able to develop networking applications with connection oriented and connectionless paradigm.
- **LO5:** Able to work with Generic classes and methods.
- **LO6:** Able to work with databases using Java.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Introduction to Java, Classes, Methods and Inheritance, Packages, Interface, Exception Handling	15	18	19
II	Streams, Multithreaded Programming, Generics, String handling	15	19	
III	Networking, Java Database Connectivity	15	19	
IV	Practical	30	19	6

Total	**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Introduction to Java: Importance of Java to the Internet (Applets and Applications),Security, Portability, Bytecode, Java Buzzwords, JAVA Programming Language Syntax, Control Statements, Arrays. **Classes, Methods and Inheritance:** Classes – Fundamentals, Declaring Objects, Methods, Constructors, this keyword, Garbage Collection, finalize() method, Overloading Methods and Constructors, Argument Passing – using objects as parameters, Returning Objects, static, final Keywords, Nested and Inner Classes. Inheritance, Using super, When Constructors Are Called, Method Overriding, Dynamic Method Dispatch, Abstract class, Using final with Inheritance; Packages, Interface. Exception Handling.

UNIT II **15 Hours**

Streams: I/O Basic (Streams, The stream classes, The predefined streams, Reading console input, writing console output, Reading and writing files), java.io Package. **Multithreaded Programming** – Thread Model, Priorities, Synchronisation, Messaging, Thread Class and Runnable Interface. Deadlock, Suspending Resuming and Stopping Threads. **Generics:** What is Generics? A Simple Generic Example. Generic with One Type Parameter, Generic with Two Type Parameters, Bounded Types, Creating Generic Methods. **String handling** – Comparison, Extraction, Various String operations, Searching strings, String and StringBuffer class.

UNIT III **15 Hours**

Networking: Socket overview, reserved sockets, Proxy servers, Internet addressing; Domain naming services (DNS), Java and the net, The networking classes and interfaces, Inet address, Factory methods, Introspection, TCP/IP server sockets, Datagrams (Datagram packet, Datagram server and client). **Java database connectivity (JDBC):** Introduction to JDBC, type of JDBC connectivity, Establishing database connections, Accessing relational databases from Java programs.

UNIT IV **30 Hours**

Practical from UNIT I, UNIT II and UNIT III.
Suggested Experiments (Appropriate JAVA SDK to be made available)

1. Write a program to create a class called Box with a parameterized constructor, along with a method to calculate the volume of the box. Use the class to find the volume of two boxes whose height, width and depth are 10, 20, 30 and 20, 30, 40 respectively.
2. Define a class called stack that can hold 10 integer values, then initialise top of the stack, with push and pop methods. Write a program to push the elements into the stack and pop out from the stack.
3. Write a Java program using a class to multiply two matrices of 3x3 order. Allow the user to input the values through the keyboard.
4. Write a Java program to find the factorial of positive integers using recursion.
5. Write a Java program to accept the command line arguments and display the arguments along with the positions.

6. Write a Java program to demonstrate method overriding where the program creates a superclass called figure that stores the dimensions of various two-dimensional objects. It also defines a method called area() that computes the area of an object. The program derives two subclasses from figure. The first is Rectangle and the second is Triangle. Each of these subclass overrides area() so that it returns the area of a rectangle and a triangle respectively.
7. Write a Java program to create a thread and start running it using Runnable interface. Allow the thread to display a message five times with a gap of 500ms.
8. Write a Java program to demonstrate the synchronisation of two threads using the synchronised statement.
9. Write a Java program to demonstrate inter thread communication considering the producer and consumer problem. There must be two classes, one for producers to produce data and another for consumers to consume data [Hint: Use wait() and nothing() to signal in both directions].
10. Write a Java program to copy the content of one file to another using java.io
11. An organisation has a record of its employees in the form of a list containing the names of employees, their date of birth, date of joining the organisation and all the designation that an employee has gone through during the tenure in the office. Write a Java program to create and maintain such a list. The list should be implemented as a vector since the number of employees is likely to grow over the years.
12. Add the following functionalities to the program written in exercise 11
    a. List all employees whose tenure in the office has been for more than 20 years
    b. The organisation has renamed the designation 'Supervisor " as"Manager ".

    Write a program to do this conversion automatically in the entire list

13. Modify the program of exercise 12 to store the data in an RDBMS of your choice. The employee list should be augmented by providing an employee id field and there should be a provision for automatically incrementing the employee id. Write a java program using jdbc to provide the following functionalities
    a. Add data
    b. View data
    c. Search data through employee id
14. Write a Java program to create a generic method that takes a list of numbers and returns the sum of all the even and odd numbers.
15. Write a socket based Java application program to create a connection between two machines such that whatever text one machine is sending to the other will be displayed at the latter’s screen and vice-versa
16. Create a Java application in which a particular machine is configured as the time server which continually listens for requests for time from clients. Clients request the server for time as a result of which the server sends the current time of the clients. The clients make a correction of the received time by adding a very small positive constant to the value and display the corrected time.

17. Implement a simple networked communications client and server. Messages are typed into the window at the server and written across the network to the client side, then they are displayed to demonstrate datagrams.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	MARKS
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. P. Naughton and H. Schildt, *"Java-2 The Complete Reference"*, McGraw Hill; 13th edition, 2021.
2. Y. D. Liang, *"Introduction to Java Programming, Comprehensive Version"*, 7th Edition, Pearson, 2019.

Reference Books:

1. D. Deitel, *"Java How To Program"*, 6th Edition, Pearson, 2004.
2. R. Krishnamoorthy and S Prabhu, *"Internet and Java Programming",* New Age International, 2002.
3. D. Flanagan, J Farley, W Crawford and K Magnusson, *"Java Enterprise in a Nutshell"*, 3rd Edition, O'Reilly, 2005.

Data Mining
(MAJOR - T + P)
BCA - 353

Paper Title: Data Mining
Paper Code: BCA-353
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Develop a comprehensive understanding of data mining concepts, including the data mining process, techniques like clustering, association rules, decision trees, and rule mining algorithms such as Apriori, Pincer-Search, and Border algorithm.
- **CO2:** Explore the principles of data warehousing, including its architecture, components, metadata, data mart, dimensional modelling, schema, and OLAP operations.
- **CO3:** Gain practical knowledge in classification and prediction techniques using KNN, MDC, Naive Bayes, Decision Tree Induction, and clustering methods such as k-means, k-medoids, PAM, CLARA, CLARANS, BIRCH, CURE, and DBSCAN.

Learning Outcomes (LOs):

On successful completion of the course, students will be able to:

- **LO1:** Ability to understand the role of data mining in the knowledge discovery process.
- **LO2:** Understand and apply a wide range of clustering, estimation, prediction, and classification algorithms, including k-means clustering, BIRCH clustering, and classification to identify patterns.
- **LO3:** To familiarise with various machine learning algorithms used in data mining.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Introduction to Data Mining and Data Warehousing.	15	18	19
II	Rule Mining	15	19	
III	Clustering, Classification and Prediction	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Data Mining: Concepts of Data Mining, Data mining process: Data preparation, data cleaning and data visualisation. KDD process. Data mining techniques: Clustering, Association rules and Decision trees. **Data Warehousing:** Overview and Concepts: Need for Data Warehousing, basic elements of Data Warehousing, difference between Database System and Data warehouse, Data Warehouse architecture and its components, Metadata, data mart, Principles of dimensional modelling, Schema, OLAP Operations.

UNIT II **15 Hours**

Rule Mining: What is an association rule? Mining association rules, frequent sets and border sets, algorithms for mining association rules - Apriori algorithm, Pincer-Search algorithm, Border algorithm. Generalised association rule, quantitative association rule, association rule with item constraint.

UNIT III **15 Hours**

Classification and Prediction: Introduction, Classification by KNN, MDC, Naive Bayes, Decision Tree Induction, Tree construction, principle, decision tree generation algorithms - CART, ID3. **Clustering:** Partitional versus Hierarchical Clustering, types of data in clustering. Partitional clustering methods - k-means, k-medoids, PAM, CLARA, CLARANS. Hierarchical clustering methods - BIRCH, CURE. Density based clustering methods- DBSCAN.

UNIT IV **30 Hours**

Practical from UNIT-I, UNIT II and UNIT-III

Suggested Experiments:

1. Importing datasets in Python (e.g., CSV files, Excel sheets).
2. Data cleaning techniques such as handling missing values, data transformation, and normalisation.
3. Data visualisation using libraries like Matplotlib and Seaborn for insights and patterns.
4. Implementing clustering algorithms like K-means or DBSCAN for grouping similar data points.
5. Association rule mining using the Apriori algorithm to discover relationships between items in a dataset.
6. Decision tree construction using libraries like Scikit-learn for classification and prediction tasks.
7. Implementing the Apriori algorithm to mine frequent itemsets and generate association rules from transactional data.
8. Building classification models using K-nearest neighbours (KNN), Naive Bayes, and Decision Trees (e.g., CART, ID3) with Python libraries like Scikit-learn.
9. Evaluating model performance using metrics such as accuracy, precision, recall, and F1-score.

10. Implementing partitional clustering algorithms (e.g., K-means, K-medoids).

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. A.K. Puzari, *"Data Mining Techniques"*, 4th Edition, University Press, 2016
2. J. Han, M. Kamber and J. Pei, *"Data Mining: Concepts and Techniques"*, 3rd Edition, Morgan Kaufman, 2011.

Reference Books:

1. P. Tan, M. Steinbach and V. Kumar, *"Introduction to Data Mining",* 1st Edition, Pearson Education, 2016.
2. J. VanderPlas, *"Python Data Science Handbook"*, 2nd Edition, O'Reilly, 2017

Semester VII

Text Analytics

(MAJOR - T + P)

BCA-401

Paper Title: Text Analytics
Paper Code: BCA-401
Number of Hours Per Week : (L+T+P= 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Understand the techniques, methods, and applications related to the analysis of textual data.
- **CO2:** Implement text analytics tasks using Python and relevant packages
- **CO3:** To inculcate knowledge on basic Natural Language Processing (NLP) concepts .

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Understand the fundamental concepts of text analytics, including NLP, information retrieval, and machine learning as applied to text.
- **LO2:** Learn various text preprocessing techniques such as tokenization, stemming, lemmatization, and stop-word removal to prepare textual data for analysis.
- **LO3:** Explore different methods for representing text data, including bag-of-words, Term Frequency-Inverse Document Frequency (TF-IDF).
- **LO4:** Explore text classification methods for assigning predefined categories or labels to text documents, including supervised learning approaches.
- **LO5:** Learn how to evaluate the performance of text analytics models using metrics like precision, recall, F1 score, accuracy.
- **LO6:** Gain practical experience by implementing text analytics algorithms and models using programming languages like Python and relevant libraries (e.g., NLTK, scikit-learn etc.)

Outline of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Overview of NLP, Python for Text Analysis	15	18	19
II	Foundations of Text Processing	15	19	
III	Text Classification, Text Similarity and Clustering	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Overview of Natural Language Processing: What is Natural Language, The Philosophy of language, Language Acquisition and Usage, Linguistics – Areas of study, Language Syntax and Structure, Words, Phrases, Clauses, Basic concept of Grammar: Dependency, Constituency, Word order typology, Language Semantics, Lexical Semantics – lemmas, word forms, homonyms, homographs and homophones, heteronyms and heterographs, polysemes, capitonyms, Synonyms and Antonyms, Hyponyms and Hypernyms, Wordnet. **Python for Text Analysis:** Working with text data – String literals, Representing strings, String Operations– Basic operations, Indexing and slicing, String Methods, Formatting, Regular expressions, Use of the text analytics framework- NLTK.

UNIT II **15 Hours**

Foundations of text processing: Text Corpora, Text processing and Normalisation - Removing HTML tags, accented characters, special characters, stemming, lemmatization, removing stopwords, Text tokenization – sentence tokenization and word tokenization, Feature Engineering for Text representation - Bag of words (BOW) model, Bag of N-grams model, implementation using scikitlearn'sCountVectorizer, TF-IDF model, Document vectorization using Tf-idf, implementation using TfidfVectorizer, Minimum edit distance.

UNIT III **15 Hours**

Text Classification: Formal definition, Automated Text classification, task variants – Binary classification, Multiclass classification, Multi-label classification, Text classification blueprint, Data preprocessing and Normalisation for Text classification, Building train and test datasets, Classification models – multinomial Naïve Bayes, logistic regression, Evaluation metrics – Accuracy, Precision, recall, F1-score, Understanding the Confusion matrix, TF-idf features with classification models. **Text Similarity and Clustering:** Text similarity, Analysing term similarity - hamming distance, Euclidean distance, cosine distance and cosine similarity, Analysing Document similarity using cosine similarity, Document Clustering: Brief description of - Hierarchical clustering models, Partitioned based or Centroid based clustering, Distribution based clustering, Density based clustering, Detailed Description of - K-Means clustering algorithm with practical implementation, Agglomerative Hierarchical Clustering.

UNIT IV **30 Hours**

Practicals for the concepts from UNIT-I, UNIT II and UNIT-III

Suggested Experiments:

1. Use regular expressions for removing :
   a) all punctuations from a body of text
   b) digits
   c) html tags
   d) white spaces
2. Compile a regular expression for words containing a sequence of two 'a's from a body of text and find the matches
3. Find all words ending in the letter 'a'
4. Perform the following string operations and methods:

concatenate two strings, indexing and slicing a string, replace(), lower(),split (),strip(),join() etc.

5. Perform lemmatization and remove all stop words from a body of text.
6. Tokenize a body of text and count the number of tokens.
7. Find the minimum edit distance between two strings
8. Using TF-IDF, find the most important words from 3 short documents.
9. Find the similarity of 3 documents using the cosine similarity measure.
10. Write a function using NLTK library to tokenize the input stream of data into words or sentences.
11. Write a Python function using regular expressions to identify noun phrases in a sentence and extract them.
12. Implement a Python script to calculate the semantic similarity between two given sentences using WordNet in NLTK.
13. Using NLTK, create a Python program that generates synonyms and antonyms for a given word. Use NLTK for accessing WordNet's lexical database.
14. Write a Python script to clean and preprocess a given text dataset, including tasks like lowercasing, punctuation removal, and stop word removal.
15. Implement a TF-IDF model using scikit-learn'sTfidfVectorizer to extract features from a text corpus.
16. Implement a spell checker in Python using dynamic programming and the minimum edit distance algorithm. Test the spell checker on a dataset of misspelt words and evaluate its accuracy.
17. Implement a Bag of N-grams model (with varying N) using scikit-learn'sCountVectorizer on a text corpus. Compare the performance of different N-gram models in capturing textual features.
18. Calculate term similarity between two given words using different distance metrics such as Hamming distance, Euclidean distance, and cosine similarity. Compare and contrast the results obtained from each metric.
19. Implement K-Means clustering algorithm from scratch in Python without using any external libraries. Apply the custom K-Means implementation to a text dataset and compare the results with scikit-learn's K-Means.
20. Implement multinomial Naïve Bayes and logistic regression classifiers using scikit-learn for a text classification task. Train the models on a labelled dataset and evaluate their performance using accuracy, precision, recall, and F1-score.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. D. Sarkar, "*Text Analytics with Python"*, Apress, 2nd Edition, 2019
2. D. Jurafsky and H. J. Martin, "*Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*." India, Prentice Hall, 2000.

Reference Books:

1. C. Manning and H.Schutze, "*Foundations of Statistical Natural Language Processing*", Cambridge, MIT Press, 1999.
2. S. Bird, et al. "*Natural Language Processing with Python*", United States, O'Reilly Media, 2009.
3. J. Kogan and M. W. Berry, "*Text Mining: Applications and Theory*" Germany, Wiley, 2010.

Theory of Computation and Compiler Design
(MAJOR - T)
BCA-402

Paper Title: Theory of Computation and Compiler Design
Paper Code: BCA-402
Number of Hours Per Week: (L+T+P = 4+0+0)
Total Contact Hours : 60
Number of Credits: 4

Course Objectives (COs):

- **CO1:** To demonstrate the interplay between different models and formal languages.
- **CO2:** Employ finite state machines to solve problems in computing.
- **CO3:** Classify machines by their power to recognize the languages.
- **CO4:** Explain deterministic and non-deterministic machines.
- **CO5:** Emphasise the concepts learnt in lexical analysis, syntax analysis, semantic analysis, intermediate code generation and type checking process through several programming exercises.
- **CO6:** To provide the understanding of language translation peculiarities by designing a complete translator for mini language.

Learning Outcomes (LOs):

On successful completion of the course, students will be able to:

- **LO1**: Employ finite state machines to solve problems in computing and classify machines by their power to recognize languages.
- **LO2:** Understand the basic concept of compiler design, and its different phases which will be helpful to construct new tools like LEX, YACC, etc.
- **LO3:** Ability to implement semantic rules into a parser that performs attribution while parsing and apply error detection and correction methods.
- **LO4:** Apply the code optimization techniques to improve the space and time complexity of programs while programming.
- **LO5:** Ability to design a compiler for a concise programming language.

Outlines of the Course:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Foundations of Automata Theory and Formal Languages	15	19	25
II	Stages of Compilation, Lexical Analysis and Parsing	15	19	

III	Semantic Analysis, Symbol Table Organization and Code Optimization	15	19	
IV	Code Generation	15	18	
Total		**60**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Languages, definitions, Regular Expressions, Regular Grammars, Acceptance of Strings and Languages, Finite Automaton Model, DFA, NFA, conversion of NFA to DFA, Conversion of Regular Expression to NFA. Regular Expressions and Regular Languages, Context-Free Grammars (CFG): Definition and designing CFGs, Derivations Using a Grammar, Parse Trees, Ambiguity and Elimination of Ambiguity, Elimination of Left Recursion, Left Factoring. Pushdown Automata and Context-Free Languages: Introduction to Pushdown Automata (PDA), Design of PDA, Equivalence of CFGs and PDAs.

UNIT II **15 hours**

Chomsky hierarchy of Languages, Phases of compilation overview, Pass, Phase, Interpretation, Bootstrapping. Top Down Parsing: Parse Trees, Ambiguous Grammars, Backtracking, LL (1), Recursive Descent parsing, Predictive parsing, pre-processing steps for predictive processing. Bottom-up parsing and handling pruning, LR (k) grammar parsing, LALR (k) grammars, Error Recovery in parsing, parsing ambiguous grammars, YACC parser generator.

UNIT III **15 hours**

Intermediate source program form: 3 address code. Introduction to Attribute Grammars, Syntax Directed Translation, Inherited Grammars, Type Checking. Symbol table format, organisation, storage allocation: static, runtime and heap allocation for arrays, strings and records. Consideration for optimization, Scope of optimization, DAG representation, Basic blocks, Common Subexpression elimination, dead code elimination.

UNIT IV **15 hours**

Absolute Code, Relocatable Machine Code, Assembler Code, Register and Address Descriptors, Implementing Global Register Allocation, Usage Counts, Using DAG for register allocation, Simple Code generation Algorithm, Generic Code generation Algorithm, Generating code from DAG.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS

	TO BE SET	TO BE ANSWERED	Marks
I	2	1	19
II	2	1	19
III	2	1	19
IV	2	1	18

Exam Duration:

THEORY
3 hours

Recommended Reading:

Text Books:

1. J. E. Hopcroft, R. Motwani and J. D. Ullman, *"Introduction to Automata Theory Languages and Computation"*, Pearson, 3rd edition, 2014
2. A.V. Aho and J D Ullman, "*Principles of Compiler Design*", Pearson Education
3. A. W. Appel, "*Modern Compiler Construction in C"*, Cambridge University Press,

Reference Books:

1. K. C. Louden, *"Compiler Construction: Principles And Practice"*, Thomson/ Delmar Cengage Learning, 2006
2. D. Brown, J. Levine and T. Mason, *"Lex&yacc"*, O'reilly Media, 2nd Edition, 2002
3. K. Cooper and L. Torczon, *"Engineering a compiler"*, O'reilly Media, 2nd Edition, Morgan Kaufmann, 2011

Internet of Things
(MAJOR / MINOR - T + P)
BCA-403 / BCA-404

Paper Title :Internet of Things
Paper Code: BCA-403 / BCA-404
Number of Hours Per Week : (L+T+P= 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Students will posses a solid understanding of the foundational concepts of IoT
- **CO2:** Students will be exposed to the interconnection and integration of the physical world and the cyber space.
- **CO3:** Students will gain proficiency in designing, developing and deploying IoT solutions on the Arduino platform.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Ability to describe and define IoT Concepts
- **LO2:** Proficiency in IoT Technologies(including sensors, actuators, microcontrollers, communication protocols (e.g., MQTT, CoAP)
- **LO3:** Skills in IoT Application Development regarding designing, developing, and deploying IoT applications for specific use cases or scenarios, integrating sensor data acquisition, data processing, decision-making logic, and actuation mechanisms.
- **LO4:** Understanding of IoT Security and Privacy
- **LO5:** Problem-Solving and Critical Thinking
- **LO6:** Effective Communication and Collaboration

Outline of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Introduction to Internet of Things (IoT) and Sensors	15	18	19
II	IoT Protocols, IoT Cloud and Securities	15	19	
III	Arduino and Node Micro Controller Unit (MCU)	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Introduction to IoT: Overview of IoT, Application areas of IoT, Characteristics of IoT, Things in IoT, IoT stack, Enabling technologies, IoT challenges, IoT levels. **Sensors:** Sensor interfacing, Types of sensors:gassensor,obstaclesensor,ultrasonicsensor,gyro sensor, LDR sensor and pH sensor, Controlling sensors, Microcontrollers

UNIT II **15 Hours**

Protocols for IoT: Messaging protocols, Transport protocols, IPv4, IPv6, UR. **Cloud for IoT**: IoT and cloud, Fog computing, Security in cloud, **IoT Security**: Security Basic, IoT system Functionalities, Security architecture, Security Requirements, Challenges in IoT Securities

UNIT III **15 Hours**

Basics of Arduino: Introduction,Arduino IDE, Basics commands, serial commands, LCD commands. **Overview of Node MCU:** Programming Node MCU using Arduino IDE, Configuring ThinkSpeak cloud Service, NodeMCU over WIFI, Sending data to ThinkSpeak cloud.

UNIT IV **30 Hours**

List of H/W components required for practical:

1. Arduino Uno Board
2. Node MCU Board
3. MQ6 and MQ125 gas sensor
4. IR sensor
5. Ultrasonic sensor
6. gyro sensor MPU-6050
7. LDR sensor
8. 16O2 LCD Display 16x2
9. Ph Sensor Kit
10. LED lights
11. 1K Ohm – 10KOhm Resistors
12. Power supply 3V-25V DC

Suggested Basic Experiments:

1. Program an LED to blink on and off at regular intervals
2. Use Push buttons to turn LED on and off
3. Read analog input from potentiometer to adjust LED brightness(using PWM)
4. Demonstrate the working of Obstacle sensors using ultrasonic sensor
5. Demonstrate the working of LDR sensor and display the output in Serial Monitor
6. Demonstrate the working of pH sensor and display the output in Serial Monitor
7. Using a GAS sensor and two LEDs (RED-present and GREEN-absent), determine if GAS is detected or not.
8. Arduino program to display hello world in LCD screen
9. Controlling an LED using LDR sensor
10. Controlling an LED using web page

Suggested Advance Experiments:

1. Connect Node MCU to the internet and send data to ThinkSpeak Cloud Platform
2. Set up MQTT communication between NodeMCU and a broker for IoT application
3. Monitor environmental parameters (eg. temperature, humidity etc) remotely via web interface.
4. Build a water sprinkler control system.
5. Air Quality Monitoring System.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	9
III	2	1	10

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient

physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. S. K. Vasudevan, A. S. Nagarajan and R. Sundaram, *"Internet of Things"* , Wiley India, 2019
2. R. Singh, A. Gehlot, L. R. Gupta, B. Singh and M. Swain, *"Internet Of Things With Raspberry Pi And Arduino"*, 2019
3. U. Dutta, N. Khurana and Devdutt, *"The Internet of Things Using NODEMCU a Practical approach to Master IoT"*, 2021

Reference Books:

1. V. Madisetti and A. Bahga, "*Internet of Things: (A Hands-on Approach*)", Universities Press (INDIA) Private Limited 2014, 1st Edition.
2. D. Hence and at el, *"IoT Fundamentals"*, Cisco Press,2017
3. R.Singh, "*IoT based Projects*", BPB, 2020

Semester VIII

Machine Learning
(MAJOR/MINOR - T + P)
BCA-450 / BCA-451

Paper Title: Machine Learning
Paper Code: BCA-450 / BCA-451
Number of Hours Per Week: (L+T+P = 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Understand the fundamental concepts of machine learning
- **CO2:** Gain proficiency in data preprocessing techniques such as data cleaning, data transformation and Exploratory data analysis (EDA)
- **CO3:** Develop skills in supervised and unsupervised learning algorithms, feature engineering and model evaluation methods

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Understand the fundamental concepts and techniques of machine learning.
- **LO2:** Learn to apply machine learning algorithms to real-world problems.
- **LO3:** Gain practical experience in implementing and evaluating machine learning models.
- **LO4:** Explore current research trends and applications of machine learning.

Outlines of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Introduction to Machine Learning Exploratory Data Analysis (EDA), Modelling Errors	15	18	19
II	Supervised Learning	15	19	
III	Unsupervised Learning, Feature Engineering and Model Evaluation and Selection.	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Introduction to ML: Definition, classification of machine learning: supervised ML, unsupervised ML, Semi-Supervised ML, Reinforcement Learning, Difference between AI and ML, Machine Learning pipeline. Data in ML**:** Labelled and unlabelled data, Numeric data, Categorical Data, Ordinal data. Overview of **Exploratory Data Analysis (EDA)**. **Data Cleaning:** missing values, outliers, noise, mean, median, mode imputation, K-Nearest

Neighbours (KNN) predictive imputation, Z-Score, Interquartile Range (IQR). **Data Transformation:** Min-Max Normalisation, Z-score Standardisation, **Encoding Categorical Data:** one-hot encoding, label encoding. **Modelling Errors**: Overview

UNIT II **15 hours**

Supervised Learning: Linear Regression: Simple and Multiple Linear Regression, cost function, Mean Squared Method, Gradient Descent, R-Mean Squared Method, Logistic Regression: Overview, Sigmoid function, K-Nearest Neighbours (k-NN), Support Vector Machines (SVM), Naïve Bayes Classifier, ID3 Algorithm, CART, Random Forests (Introduction), Neural Networks (Introduction).

UNIT III **15 hours**

Unsupervised Learning: Clustering in ML, Types of clustering methods, Distance Metrics: Euclidean Distance, Cosine Similarity, K-Means Clustering, DBSCAN algorithm, Hierarchical Clustering, Hidden Markov Model (HMM). **Feature Engineering:** Overview, polynomial features, feature scaling, recursive feature elimination, Principal Component Analysis (PCA), t-Distributed Stochastic Neighbour Embedding (t-SNE), Association Rule Learning (Apriori). **Model Evaluation and Selection:** Cross-validation, Bias-Variance Trade off, Confusion Matrix, Evaluation Metrics (e.g., Accuracy, Precision, Recall, F1 Score).

UNIT IV **30 hours**

Suggested Experiments:

1. Use pandas to load a dataset into a DataFrame and display the first few rows.
2. Use pandas to get a summary of the dataset, including information about the columns and data types.
3. Identify and handle missing data by either imputing missing values or removing rows/columns with missing values.
4. Clean the data by removing duplicates, correcting data types, and standardising values.
5. **Data Visualization**:
    a. Create a histogram of a numerical variable to visualise its distribution.
    b. Create a box plot to visualise the distribution of a numerical variable and identify outliers.
    c. Create a scatter plot to visualise the relationship between two numerical variables.
    d. Create a bar plot to visualise the frequency of a categorical variable.
6. Use seaborn to create a heatmap of the correlation matrix to visualise the relationships between numerical variables.
7. Use pandas to group the data by a categorical variable and calculate summary statistics for each group.

8. **Feature Engineering**:
    a. Create new features by transforming existing ones (e.g., extracting date components from a datetime column).
    b. Encode categorical variables using one-hot encoding or label encoding.
9. **Data Transformation**:
    a. Normalise numerical variables to scale them to a standard range.
    b. Standardise numerical variables to have a mean of 0 and a standard deviation of 1.
10. **Spam Email Classification using Naive Bayes Classifier:**
    a. Use a dataset containing emails labelled as spam or not spam.
    b. Preprocess the data by tokenizing the emails, removing stop words, and converting words to lowercase.
    c. Implement a Naive Bayes classifier to classify emails as spam or not spam based on their content.
11. Implement linear regression using NumPy on a dataset like the Boston Housing dataset to predict house prices.
12. Use scikit-learn to implement linear regression on a real dataset and evaluate its performance.
13. Use scikit-learn to implement logistic regression for binary classification on a dataset like the Titanic dataset.
14. Implement logistic regression from scratch using NumPy and compare the results with scikit-learn.
15. Implement decision trees from scratch using Python and use them for classification on a dataset like the Iris dataset.
16. Use scikit-learn to implement random forests for classification on a dataset like the Titanic dataset.
17. Use scikit-learn to implement SVM for classification on a dataset like the Iris dataset.
18. Implement SVM from scratch using a library like CVXOPT and apply it to a simple dataset for binary classification.
19. Implement k-NN from scratch using NumPy and apply it to a dataset like the Iris dataset for classification.
20. Use scikit-learn to implement k-NN for classification on a dataset like the Titanic dataset.
21. Use scikit-learn to implement K-Means clustering on a dataset like the Iris dataset to cluster the data into distinct groups.
22. Apply K-Means clustering to a dataset for image compression.
23. Use scikit-learn to apply PCA to reduce the dimensionality of a dataset like the Iris dataset and visualise the principal components.
24. Use PCA for feature extraction and apply a machine learning model on the transformed dataset.
25. Use scikit-learn to implement cross-validation to evaluate the performance of different machine learning models on a dataset.
26. Compare the performance of models using different evaluation metrics (e.g., accuracy, precision, recall).

27. Experiment with different feature engineering techniques (e.g., polynomial features, feature scaling) using scikit-learn and observe their impact on model performance.
28. Use scikit-learn to implement feature selection techniques (e.g., recursive feature elimination) to identify the most important features for a given task.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical:

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode and algorithm)

20% : Error Trapping

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. A. Geron, *"Hands on Machine Learning with Scikit-Learn &Tensorflow"*, 3rd Edition, O'Reilly, 2017.
2. C.M. Bishop, "*Pattern Recognition and Machine Learning"*, Reprint-First Edition 2006, Springer, 2016
3. J. VanderPlas, *"Python Data Science Handbook"*, 2nd Edition, O'Reilly, 2017

Reference Books:

1. M. Andreas, *"Introduction to Machine Learning with Python"*, O'Reilly, 2016
2. W. McKinney, *"Python for Data Analysis"*, 2nd Edition, O'Reilly, 2017
3. S. Raschka, "Machine Learning with Scikit-Learn", Packt Publishing, 1st Edition, 2022

Computer Oriented Numerical Methods
(MAJOR - T + P)
BCA-453

Paper Title: Computer Oriented Numerical Methods
Paper Code: BCA-453
Number of Hours Per Week: (L+T+P = 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3 + Pr: 1)

Course Objectives (COs):

- **CO1:** Introduce fundamental concepts in Computer Arithmetic and Equation Solving,
- **CO2:** Introduce Polynomial Interpolation, Numerical Differentiation, Integration, and Differential Equations.
- **CO3:** Implement the above algorithms using the C language.

Learning Outcomes (LOs):
On successful completion of the course, students will be able to:

- **LO1:** Demonstrate proficiency in Computer Arithmetic and Equation Solving.
- **LO2:** Utilise Polynomial Interpolation Techniques to interpolate data.
- **LO3:** Applying Numerical Techniques for Differentiation and Integration
- **LO4:** Solving Differential Equations Using Numerical Methods
- **LO5:** Developing Computational Skills in Mathematical Analysis

Outlines of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Computer Arithmetic,Solution of a Single Polynomial or Transcendental Equation,Solution of simultaneous algebraic Equations.	15	18	19
II	Polynomial Interpolation	15	19	
III	Numerical Differentiation and Integration, Solution of Differential Equations	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 hours**

Computer Arithmetic: Normalised floating-point representation of real numbers and

operations using it; normalisation and its consequences. Errors in Arithmetic Operations: Types and measurement, absolute and relative error, approximation and significant figure. **Solution of a Single Polynomial or Transcendental Equation**: Rate of convergence of iterative methods (definition only), Method of bisection, false-position, Newton Raphson method, secant method, comparison of the methods. **Solution of simultaneous algebraic Equations:** Gauss elimination method, pivotal condensation, ill conditioned equations and iterative refinement, Gauss-Seidel iterative method.

UNIT II **15 hours**

Polynomial Interpolation: Lagrange's interpolating polynomial, difference tables and Newton's divided difference interpolating polynomial, Newton-Gregory forward and backward difference interpolating polynomials.

UNIT III **15 hours**

Numerical Differentiation and Integration: numerical differentiation, quadrature formulae, trapezoidal rule, and Simpson's one-eight rules. **Solution of Differential Equations:** Euler's method, second and fourth order Runge-Kutta methods, predictor-corrector method for solving first order, first-degree differential equations.

UNIT IV **30 hours**

Practical Assignments (Any Appropriate C compiler to be made available)

Suggested Experiments:

1. Write a program to find the solution of any polynomial or transcendental equations by using Bisection Method
2. Write a program to find the solution of any polynomial or transcendental equations by using the false-position method.
3. Write a program to find the solution of any polynomial or transcendental equations by using Newton-Raphson method.
4. Write a program to find the solution of any polynomial or transcendental equations by using secant method.
5. Write a program to find the solution of simultaneous algebraic equations by Gauss elimination method.
6. Write a program to compute an interpolation by Newton’s divided difference interpolating polynomial.
7. Write a program to compute an interpolation by Newton-Gregory forward difference interpolating polynomials.
8. Write a program to compute an interpolation by Newton-Gregory backward difference interpolating polynomials.
9. Write a program to solve a numerical integration by trapezoidal rule.
10. Write a program to solve a numerical integration by Simpson’s one-third rule.
11. Write a program to solve a numerical integration by Simpson’s three-eight rule.
12. Write a program to find the derivatives using backward difference formulas.
13. Write a program to find the derivatives using the forward difference formula.
14. Write a program to solve any differential equation by using Euler's method.

15. Write a program to solve any differential equation by using second and fourth order Runge-Kutta methods.
16. Write a program to solve any first order, first-degree differential equation by using predictor-corrector method .

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical:

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode and algorithm)

20% : Error Trapping

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. V. Rajaraman, *"Computer Oriented Numerical Methods"*, 5th Edition, Prentice Hall India, New Delhi, 2002.
2. B. S. Grewal, *"Numerical Methods"*, 7th Edition, Khanna Publishers, Delhi, 2005

Reference Books:

1. E. Balagurusamy, "*Numerical Methods*", Standard Edition, McGraw Hill Education, 2017
2. S. C. Chapra and R. P. Canale, *"Numerical Methods for Engineers"*, 8th Edition, McGraw Hill, 2021

Mobile Application Development

(MAJOR - T + P)
BCA - 454

Paper Title: Mobile Application Development
Paper Code: BCA-454
Number of Hours Per Week: (L+T+P= 3+0+2)
Total Contact Hours: 75
Number of Credits: 4 (Th: 3+ Pr: 1)

Course Objectives (COs):

- **CO1:** Introduce mobile applications development to the students using the android platform as the basis of teaching the techniques and
- **CO2:** Introduce UI/UX application concepts.
- **CO2:** Introduce Design patterns and equipping students with the essential knowledge and skills to design, develop, and deploy Android applications.

Learning Outcomes (LOs):

- **LO1:** Describe and define android programming structure
- **LO2:** Extend one knowledge in application development
- **LO3:** Obtain new findings through experiment and demonstration
- **LO4:** Design and develop their own application which is tailored to their needs
- **LO5:** Create and formulate plans and solutions that will help solve problems
- **LO6:** Integrate the skills with the ability to access and decide self-selected criteria based on observation.

Outline of the Course:

UNIT	TOPIC	Hours	External Marks	Internal Marks
I	Introduction	15	18	19
II	User Interaction and Experience	15	19	
III	Saving User Data	15	19	
IV	Practical	30	19	6
Total		**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Introduction to Android: Your first Android app, Layouts and resources for the UI. Activities and intents: Activities and intents, Activity lifecycle and state. Testing, debugging, and using support libraries: The Android Studio debugger, The Android Support Library

UNIT II **15 Hours**

User Interaction: Buttons and clickable images, Input controls, Menus and pickers, User navigation. **User experience:**Drawables, styles, and themes ,Material Design , Resources for adaptive layouts AsyncTask and AsyncTaskLoader, Connecting to the Internet, Broadcast receivers, Services, Notifications, Alarm managers, Transferring data efficiently

UNIT III **15 Hours**

Preferences and SQLite: Data storage with SQLite ,Sharedpreferences,SQLite primer, store data using SQLite database,ContentProviders, loaders to load and display data, Permissions, performance and security.

UNIT IV **30 Hours**

Practical involved the concepts from UNIT-I, UNIT II and UNIT-III
Suggested Experiments:

1. Install Android Studio and Run Hello World Program.
2. Create an android app with Interactive User Interface using Layouts.
3. Create an android app that demonstrates working with TextView Elements.
4. Create an android app that demonstrates Activity Lifecycle and Instance State.
5. Create an android app that demonstrates the use of Keyboards, Input Controls, Alerts, and Pickers.
6. Create an android app that demonstrates the use of an Options Menu.
7. Create an android app that demonstrates Screen Navigation using the App Bar and Tabs.
8. Create an android app to Connect to the Internet and use BroadcastReceiver.
9. Create an android app to show Notifications and Alarm manager.
10. Create an android app to save user data in a database and use of different queries.

Suggested advance experiments:

1. Create a calculator app with all the basic features.
2. Create a note app where users can note down important information and store the data in the database.
3. Develop an app that uses AsyncTask to perform background tasks, such as fetching data from the internet.
4. Build an app that fetches data in JSON format from an API and parses it to display relevant information in the app.
5. Integrate a WebView component to display web content within the app.
6. Integrate a library like Picasso or Glide to efficiently load and cache images in the app.

7. Develop an app with multiple screens and practice navigating between them using intents.
8. Build a basic currency converter app that converts between two currencies.
9. Develop a simple quiz app with multiple-choice questions and track user scores.
10. Build a simple phone book app that will store the contact in the database.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	TO BE ANSWERED	Marks
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. M. K. C. Yadav, P. S. Palkar and R. Jaiswal, *"Android Developer Fundamentals"*, 1st Edition, Himalaya Publishing House, 2018.
2. D. Griffiths, *"Head First Android Development: A Brain-Friendly Guide"*, 2nd Edition, O'Reilly, 2017

Reference Book:

1. J. Horton, *"Android Programming For Beginners"*, 3rd Edition, Packt Publishing, 2015.
2. J. Sheusi, *"Android Application Development for JAVA Programmers"*, 1st Edition, Cengage, 2013

Artificial Intelligence

(MAJOR - T + P)

BCA-455

Paper Title: Artificial Intelligence
Paper Code: BCA-455
Number of Hours Per Week : (L+T+P= 3+0+2)
Total Contact Hours : 75
Number of Credits: 4 (Th: 3+ Pr: 1)

Course Objectives (COs):

- **CO1:** Understand the fundamental concepts and techniques of artificial intelligence.
- **CO2:** Identify problems that may be solved using artificial intelligence.
- **CO3:** Understand knowledge representation using logic and reasoning
- **CO4:** Understand the concepts of machine learning and its algorithms.
- **CO5:** Apply basic principles of AI in solutions that require problem-solving, inference, perception, knowledge representation, and learning.

Learning Outcomes (LOs) :
On successful completion of the course, students will be able to:

- **LO1:** Understand the informed and uninformed problem types and apply search strategies to solve them.
- **LO2:** Apply difficult real-life problems in a state space representation so as to solve them using AI techniques like searching and game playing.
- **LO3:** Design and evaluate intelligent expert models for perception and prediction from an intelligent environment.
- **LO4:** Formulate valid solutions for problems involving uncertain inputs or outcomes by using decision making techniques.
- **LO5:** Examine the issues involved in knowledge bases, reasoning systems and planning.
- **LO6:** Understand and implement the procedures for machine learning algorithms

Outline of the Paper:

UNIT	Topic	Hours	External Marks	Internal Marks
I	Overview and Problem Solving Techniques	15	18	19
II	Knowledge Representation using Logic and Reasoning	15	19	
III	Introduction to Machine Learning	15	19	
IV	Practical	30	19	6

Total	**75**	**75**	**25**

CONTENTS

UNIT I **15 Hours**

Overview and Problem Solving Techniques: Definition of AI, foundations of AI, AI technique, State space, defining a problem as a state space, Uninformed search strategies - Breadth-First, Depth-First, Iterative deepening, Heuristic functions, Heuristic search strategies (Best-First Search, A*, Hill climbing search, Steepest Ascent Hill climbing), Problem Reduction search: AO* algorithm, Constraint satisfaction problems, Game Playing (Overview, Minimax algorithm, alpha-beta pruning)

UNIT II **15 Hours**

Knowledge Representation using Logic and Reasoning: Knowledge representation using logic, representing facts in logic, representing instances and ISA relationships, computable functions and predicates, resolution, conversion to clausal form, basis of resolution, resolution in propositional logic, unification, resolution in predicate logic, question answering, Representation of knowledge using rules: Procedural vs. Declarative Knowledge, Logic Programming, Forward vs. Backward Reasoning, Matching, Control Knowledge.

UNIT III **15 Hours**

Introduction to Machine Learning : Machine Learning basics, Applications of ML,Data Mining Vs Machine Learning vs Big Data Analytics. **Supervised Learning:** Naïve Base Classifier, Classifying with k-Nearest Neighbour classifier, Decision Tree classifier. **Unsupervised Learning:** Grouping unlabeled items using k-means clustering. **Association rule mining:** Analysis with the Apriori algorithm Introduction to reinforcement learning.

UNIT IV **30 Hours**

Suggested Basic Experiments:

1. Implementation of Breadth First Search using Python.
2. Implementation of Depth First Search using Python.
3. Implementation of Hill Climbing Algorithm.
4. Implementation of A* Algorithm.
5. Implementation of AO* Algorithm.
6. Implementation of Tic-Tac-Toe game using Python.
7. Implementation of an 8-Puzzle problem using Python.
8. Implementation of the Water-Jug problem using Python.
9. Implementation of Travelling Salesman Problem using Python.
10. Implementation of Tower of Hanoi using Python.
11. Implementation of Alpha-Beta Pruning using Python.
12. Implementation of Linear Regression.
13. Implementation of Logistic regression.
14. Implementation of Decision tree classification.
15. Implementation of Naïve Bayes classifier algorithm.

16. Implementation of K-nearest Neighbour.
17. Implementation of K-means Clustering.

Suggested Advance Experiments:

1. Write a function to perform BFS on a given graph represented as an adjacency matrix or adjacency list.
    a. Implement BFS to find the shortest path between two nodes in a graph.
    b. Visualise the traversal path of BFS on a graphical representation of a maze.
2. Implement DFS using recursion to traverse a binary tree and print out the nodes in preorder, inorder, and postorder traversal.
    a. Write a function to perform DFS on a given graph and return the path from the start node to a goal node.
    b. Develop a program to generate and visualise a maze, and then use DFS to find a path from the start to the exit.
3. Write a program to implement Iterative Deepening Search (IDS) to solve the Eight Queens Puzzle.
    a. Implement IDS to find the shortest path between two nodes in a graph represented as an adjacency matrix.
    b. Compare the performance of BFS, DFS, and IDS on large graphs in terms of time and memory usage.
4. Write a function to implement the A* search algorithm to find the shortest path between two points on a map.
    a. Implement A* search to solve the N-puzzle problem (Sliding Puzzle).
5. Write a Constraint Satisfaction Problems (CSP) solver to solve a Sudoku puzzle using backtracking with constraint propagation.
    a. Implement a CSP solver to solve the N-Queens problem using the Minimum Remaining Values (MRV) heuristic and Forward Checking.
6. Write a program to implement the Minimax algorithm with alpha-beta pruning to play Tic-Tac-Toe against an AI opponent.
7. Write a python program to implement a Naïve Bayes classifier to classify emails as spam or non-spam. Train the classifier using a labelled dataset and evaluate its performance on a separate test dataset.
8. Develop a python script to classify handwritten digits from the MNIST dataset using the k-Nearest Neighbors classifier. Experiment with different values of k and evaluate the classifier's accuracy.
9. Implement a decision tree classifier in Python to predict the species of iris flowers using the famous Iris dataset. Train the classifier and visualise the resulting decision tree.
10. Write a Python program to perform k-means clustering on a dataset containing unlabeled items (e.g., customer data). Visualise the clusters and analyse their characteristics.

11. Develop a Python script to mine association rules from a transaction dataset (e.g., market basket analysis) using the Apriori algorithm. Display the generated association rules along with their support and confidence values.
12. Implement a simple reinforcement learning environment in Python, such as a grid-world navigation problem. Use Q-learning to train an agent to navigate from a starting point to a goal while avoiding obstacles.

Instructions to Paper Setter (Theory Examination):

- Questions should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	18
II	2	1	19
III	2	1	19

Instructions to Paper Setter (Practical Examination):

- Two sets of question papers should be set according to the following scheme

UNIT	QUESTIONS		
	TO BE SET	**TO BE ANSWERED**	**Marks**
I	2	1	6
II	2	1	6
III	2	1	7

Exam Duration:

THEORY	PRACTICAL
$2\frac{1}{2}$ hours	3 hours

Distribution of marks for practical

10% : Syntax and input/output screens

30% : Logic and efficiency (source code, pseudocode, and algorithm)

20% : Error trapping (illegal or invalid input, stack overflow, underflow, insufficient physical memory, etc

20% : Completion

20% : Result

Recommended Reading:

Text Books:

1. S. J. Russell and P. Norvig, "*Artificial Intelligence: A Modern Approach",* 4th Edition, Pearson Education Inc., 2022
2. E.Rich, K. Knight and S.B. Nair, "*Artificial Intelligence"*, 3rd Edition, Tata McGraw-Hill, 2017

Reference Books:

1. G. F. Luger, "*Artificial Intelligence: Structures and Strategies for Complex Problem Solving",* 6th Edition, Pearson Education Inc., 2021
2. N. Walkins, *"Artificial Intelligence: The Ultimate Guide to AI, The Internet of Things, Machine Learning, Deep Learning + a Comprehensive Guide to Robotics"*, Amazon Digital Services LLC, 2019
3. M. Negnevitsky, *"Artificial Intelligence: A Guide to Intelligent Systems"*, 2nd Edition, New Delhi: Pearson Education Inc., 2002