DEBRA THANA SAHID KSHUDIRAM SMRITI MAHAVIDYALAYA

Gangaram Chak, Chak Shyampur, Debra, West Bengal

PROPOSED SYLLABUS (DRAFT) OF

MAJOR COURSE UNDER CCFUP, 2023 FOR SEMESTER–I & II

FOR COMPUTER SCIENCE(MAJOR) PROGRAMMES

(w.e.f. Academic Year 2024-2025)

Based on

Curriculum & Credit Framework for Undergraduate Programmes (CCFUP), 2023 & NEP, 2020

Level	YR.	SEM	Course Type	Course Code	Course Title	Credit	L-T-P	Marks		
								CA	ESE	TOTAL
B.Sc. (Hons.)	1st	I	SEMESTER-I							
			Major-1	UG/I/COMP/4/MJ-1	**T: Introduction to Programming in C** **P: Programming in C Lab**	4	3-0-1	15	60	75
			SEC	UG/I/COMP/4/SE-1	**P: Web design using HTML and CSS**	3	0-0-3	10	40	50
			AEC	AEC 01	Communicative English -1 (***common for all programmes***)	2	2-0-0	10	40	50
			MDC	MDC 01	Multidisciplinary Course -1 (***to be chosen from the list***)	3	3-0-0	10	40	50
			VAC	VAC 01	ENVS (***common for all programmes***)	4	2-0-2	50	50	100
			Minor (Disc.-I)	UG/I/COMP/4/MI-1	**T: Computer Fundamental** **P: Office Automation** ***(To be taken by students of other Disciplines)***	4	3-0-1	15	60	75
					Semester-I Total	20				400
		II	SEMESTER-II							
			Major-2	UG/II/COMP/4/MJ-2	**T: Digital Logic Design** **P: Digital Logic Design Lab**	4	3-0-1	15	60	75
			SEC	UG/II/COMP/4/SE-2	**P: Problem Solving Using Python**	3	0-0-3	10	40	50
			AEC	AEC 02	MIL-1 (***common for all programmes***)	2	2-0-0	10	40	50
			MDC	MDC 02	Multi-Disciplinary Course-02 (***to be chosen from the list***)	3	3-0-0	10	40	50
			VAC	VAC 02	Value Added Course-02 (***to be chosen from the list***)	4	4-0-0	10	40	50
			Minor (Disc.-II)	UG/II/COMP/4/MI-2	**T: Introduction to Programming using C** **P: Programming in C Lab** ***(To be taken by students of other Disciplines)***	4	3-0-1	15	60	75
			Summer Intern.	CS	**Community Service**	4	0-0-4	-	-	50
					Semester-II Total	24				400
					TOTAL of YEAR-1	44				800

Semester I

MAJOR 1

UG/I/COMP/4/MJ-1: Introduction to Programming in C Credits 04 (Full Marks: 75)

Course Objective:

- Attain a thorough comprehension of programming paradigms encompassing procedural, object-oriented, and functional styles, elucidating their pragmatic utilization in resolving tangible challenges.
- Cultivate adeptness in problem-solving methodologies, encompassing algorithmic cogitation, problem dissection, and pseudocode articulation, to adeptly confront computational quandaries.
- Garner an exhaustive comprehension of pivotal algorithms like searching, sorting, and recursion, and scrutinize their efficiency concerning temporal and spatial complexity.
- Attain proficiency in rudimentary C programming language tenets, comprising syntax intricacies, data type nuances, operator intricacies, and control flow statement nuances, fostering the composition of efficient and structured code.
- Assimilate the art of adroitly manipulating arrays and strings within C, leveraging string handling functions to fulfill sundry tasks with dexterity.
- Mastery of functions and modular programming paradigms within C, including the nuances of parameter passing mechanisms and the merits of modular programming, culminating in a profound command over these fundamental concepts.
- Grasp the intricacies of variable scope and storage class intricacies within C, facilitating the distinction between local, global, and static variables with precision.
- Delve into the nuances of structures and pointers within C, encompassing structural declaration intricacies, pointer fundamentals, and their multifarious applications within programming realms.
- Cultivate adeptness in file handling operations within C, spanning the spectrum from opening, reading, writing, and closing files, to a nuanced comprehension of file pointers and their manifold operations.
- Effectuate a seamless transition into the realm of C++ programming language, attaining prowess in sophisticated concepts such as operator overloading, inheritance, polymorphism, and exception handling, to fashion robust and efficacious software solutions.

UG/I/COMP/4/MJ-1(T): Introduction to Programming in C (45 Hours) **Credits 03**

Course Outline:

1. **Introduction to Programming (3 Hours)**
    - Introduction to various programming styles like procedural, object-oriented, and functional.
    - Understanding the unique features and practical applications of each style.
    - Examining the role of programming in addressing real-world challenges.
    - Fundamentals of Problem Solving.
    - Introduction to Symbolic Constant, Pre-Processor Directives and Header Files.
    - Understanding the process of analyzing and deconstructing problems, including identifying inputs and output.
    - Basics of thinking algorithmically and expressing ideas using pseudocode and Flowchart.
    - Introduction to essential algorithms such as searching, sorting, and recursion.
2. **Basics of C Language (7 Hours)**
    - Overview of the syntax and structure of the C programming language.
    - Understanding different types of variables, data types, and operators in C.
    - Learning basic input and output operations in C.
    - Exploring control flow statements like if-else, nested if-else, switch-case, break, continue and goto.
    - Learning about various loop structures like for loop, while loop, and do-while loop.
    - Applying conditional statements and loops in practical scenarios with examples.
3. **Arrays and Strings (7 Hours)**
    - Introduction to arrays and their manipulation techniques in C.
    - Exploring string handling functions like strcpy, strcat, strlen, etc.
    - Practical exercises focusing on array manipulation and string handling.
4. **Functions and Modular Programming (7 Hours)**
    - Basics of functions including declaration, definition, and invocation.
    - Understanding parameter passing mechanisms such as by value and call by references, Recursive functions, Macro function.
    - Exploring the principles and advantages of modular programming.
    - Implementing modular programs through practice sessions.
5. **Storage Classes and Scope (3 Hours)**
    - Exploring variable scope concepts including local, global, and static.

- Overview of storage classes such as auto, extern, static, and register with example.

6. **Structure and Union (6 Hours)**
    - Basics of structure declaration and initialization.
    - Understanding pointer fundamentals including declaration, dereferencing, and arithmetic operations.
    - Exploring the applications of structures and pointers in programming.

7. **Pointers (6 Hours)**
    - Discussion on memory address operators, explanation of declaring pointer types.
    - Assignment and initialization of pointers, performing arithmetic operations using pointers.
    - Functions that utilize pointers.
    - Exploring the connection between arrays and pointers
    - Handling pointers within structures.
    - Dynamic management of memory.

8. **File Processing (6 Hours)**
    - Exploring file handling operations like opening, reading, writing, and closing files.
    - Understanding file pointers and their operations.
    - Practical exercises focusing on file handling and processing.

UG/I/COMP/4/MJ-1(P): Programming in C Lab **Credits: 01**

1. Write a program to check a year is Leap year or not.
2. Write a program to solve the following Quadratic equation $Ax^2 + Bx^2 + C = 0$
3. Write a program to print the sum and product of digits of an integer.
4. Write a program to find the reverse a number and then check the number is palindrome or not.
5. Write a program to compute the sum of the first n terms of the following series
   S = 1+1/2+1/3+1/4+……
6. Write a program to compute the sum of the first n terms of the following series
   S =1-2+3-4+5…….

7.Write a program to find the value of cosX from the following Cos series:

CosX = 1-x2/2!+x4/4!-.....∞

8.Write a program to find the GCD and LCM of two numbers.

9.Write a program to display Armstrong numbers between the range a to b.

10. Write a program to display Strong numbers between the range a to b.

11. Write a program to convert a Decimal number into its equivalent Binary number.

12. Write a program to convert a Binary number into its equivalent Decimal number.

13. Write a program to convert a Binary number into its equivalent Octal number.

14. Write a function to find whether a given no. is prime or not. Use the same to generate the prime numbers less than 100.

15. Write a program to compute the factors of a given number.

16. Write a function that checks whether a given string is Palindrome or not. Use this function to find whether the string entered by user is Palindrome or not.

17. Write a program to count number of vowels, consonants, digits and blank spaces in a line of text.

18. Write a macro that swaps two numbers. WAP to use it.

19. Write a program in which a function is passed address of two variables and then alter its contents.

20. Write a program to print a triangle of stars as follows (take number of lines from user):

```
*
***
*****
*******
*********
```

21. Write a program to print the pyramid of numbers as follows (take number of lines from user):

```
1
121
12321
1234321
123454321
```

22. Write a program to display Fibonacci series (i) using recursion, (ii) using iteration

23. Write a program to calculate Factorial of a number (i) using recursion, (ii) using iteration

24. Write a program to calculate GCD of two numbers (i) with recursion (ii) without recursion.

25. Write a program to perform following actions on an array entered by the user:
    i) Print the even-valued elements
    ii) Print the odd-valued elements
    iii) Calculate and print the sum and average of the elements of array
    iv) Print the maximum and minimum element of array
    v) Remove the duplicates from the array vi) Print the array in reverse order
26. Write a program to arrange the list of n numbers in ascending order.
27. Write a program that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.
28. Write a program which takes the radius of a circle as input from the user, passes it to another function that computes the area and the circumference of the circle and displays the value of area and circumference from the main() function.
29. Write a menu driven program to perform following operations on strings:
    a) Show address of each character in string
    b) Concatenate two strings without using strcat() function.
    c) Concatenate two strings using strcat() function.
    d) Compare two strings.
    e) Calculate length of the string (use pointers).
    f) Convert all lowercase characters to uppercase.
    g) Convert all uppercase characters to lowercase.
    h) Calculate number of vowels.
    i) Reverse the string.
30. Create a structure Student containing fields for Roll No., Name, Class, Year and Total Marks. Create 10 students and store them in a file.
31. Write a program to retrieve the student information from file created in previous question and print
    it in following format: Roll No. Name Marks
32. Copy the contents of one text file to another file, after removing all whitespaces.
33. Write a program that will read 10 integers from user and store them in an array. Implement array using pointers. The program will print the array elements in ascending and descending order.

Reference Books

For C Programming Language

1. **Byron S Gottfried "Programming with C", Fourth edition, Tata McGrawhill.**
2. **E. Balagurusamy, "Programming with ANSI-C", Eight Edition,2008, Tata McGraw Hill.**
3. **Kanetkar Y, "Let us C", BPB Publications, 2017.**
4. **Brian W. Kernighan & Dennis Ritchie, "The C Programming Language", Second Edition, Pearson.**
5. **Y. Kanetkar, "Understanding Pointers in C", Paperback.**

SEC 1

UG/I/COMP/4/SE-1(P): Web design using HTML and CSS — Credits: 03

Course Objective:

- Develop a comprehensive understanding of key web technologies and the client-server architecture.
- Acquire proficiency in HTML and CSS, exploring diverse tags, elements, and the fundamental structure of HTML documents.
- Master the art of effective webpage styling using CSS, including selectors, properties, and layout techniques.
- Grasp the principles of responsive web design, mobile optimization, and media query implementation.
- Explore advanced features in HTML, including HTML5 elements, and delve into advanced CSS concepts and best practices.
- Create interactive forms, implement animations and transitions, and explore pseudo-classes and pseudo-elements using HTML and CSS.
- Recognize and implement web accessibility best practices, utilizing ARIA roles and attributes.
- Introduce and utilize CSS preprocessors.
- Deepen the ability to create advanced responsive layouts with CSS, exploring intricate layout techniques and media query applications.
- Familiarize themselves with popular CSS frameworks like Bootstrap, integrating pre-built components and styles into web projects.

Course Outline:

1. **Introduction to Web Development**
   - Overview of web technologies.
   - Client-server architecture.
   - Introduction to HTML and CSS.
2. **HTML Fundamentals**

- Program Explore HTML tags and elements.
- Understand the document structure in HTML.
- Learn about forms, multimedia, and semantic HTML.

3. **CSS Styling Techniques**
    - Learn CSS for styling web pages.
    - Explore CSS selectors and properties.
    - Understand layout techniques.
4. **Advanced HTML and CSS**
    - Explore advanced HTML features.
    - Learn about HTML5 and its new elements.
    - Dive into advanced CSS concepts and best practices.
5. **Interactive Elements with HTML and CSS**: **(5 Hours)**
    - Create interactive forms using HTML.
    - Implement animations and transitions with CSS.

HTML and CSS Practical:

1.Create a simple webpage with headings, paragraphs, and a list.
2.Design a form with various input types (text, password, radio buttons, and checkboxes).
3.Build a table displaying information with proper headers and rows.
4.Implement an ordered and unordered list to showcase a set of items.
5.Develop a webpage with hyperlinks linking to different sections within the same page.
6.Design a responsive navigation bar with dropdown menus.
7.Create an HTML page that includes multimedia elements such as images, audio, and video.
8.Develop a form that utilizes HTML5 semantic elements (e.g., <article>, <section>).
9.Construct a simple HTML5 canvas drawing with basic shapes.
10. Implement a webpage with an embedded Google Map.
11. Style a webpage using internal CSS to change fonts, colors, and background.
12. Create a CSS file and link it to an HTML file for external styling.
13. Design a responsive layout using Flexbox for better alignment.
14. Use CSS Grid to create a two-dimensional layout with rows and columns.

15. Implement CSS transitions for smooth effects on hover or click events.
16. Style a form with CSS to enhance its visual appeal.
17. Customize the appearance of hyperlinks with different states (normal, hover, visited).
18. Create a CSS animation for a specific element on your webpage.
19. Style a navigation bar to have a fixed position when scrolling.
20. Develop a webpage that replicates a login/signup form with proper validation.
21. Create a responsive landing page with a hero section and call-to-action buttons.
22. Design a pricing table using HTML for structure and CSS for styling.
23. Build a card layout displaying information with images, titles, and descriptions.
24. Implement a dropdown menu in the navigation bar using HTML and CSS.

MINOR 1

UG/I/COMP/4/MI-1: Computer Fundamental (Credits 04) [Full Marks: 75]

Course Objective:

- Provide students with a thorough grasp of the fundamental principles, components, and operations of computers.
- Equip students with the essential knowledge and skills to proficiently use computers in both personal and professional contexts.
- Familiarize students with key computer concepts and terminology.
- Foster an understanding of the historical evolution of computers and the various generations of computer technology.
- Develop insights into the basic structure and operations of a computer system.
- Instruct students on the vital hardware components, including the CPU, memory, storage devices, input devices, and output devices
- Illuminate the purpose and functionality of each hardware component.

UG/I/COMP/4/MI-1(T): Computer Fundamental (45 Hours) **Credits 03**

Course Outline:

1. **Introduction (3 Hours)**
    - Define computer and discuss its characteristics.
    - Explore the generations of computers and their classifications (Micro, Mini, Mainframe, Super).
    - Examine applications of computers and introduce basic concepts of software and hardware.
    - Cover fundamental notions such as Bit, Byte, Word, Nibble, and various computer languages.

2. : **Basic Components of Computer (7 Hours)**
    - Discuss the basic organization of a digital computer, including CPU, CU, ALU, Register set, and Communication Pathway.
    - Provide a basic explanation of CPU components, such as CU, ALU, and Register set.
    - Define Communication Pathway, covering aspects like Bus, Internal and External Bus, Control, Address, and Data Bus.
    - Explore input and output devices, including keyboards, pointing devices, and Memory hierarchy.

3. **Number System (10 Hours)**
    - Define positional and non-positional number systems, including Binary, Decimal, Octal, and Hexadecimal.
    - Explore binary-decimal-octal-hexadecimal arithmetic, signed and unsigned numbers, and complement notation.
    - Cover addition and subtraction operations using complement notation and floating-point representation of numbers.
    - Discuss computer codes, including weighted binary codes, non-weighted binary codes, and alphanumeric codes.

4. **Data Communication and Computer Network (10 Hours)**
    - Define data communication, its characteristics, components, and modes.
    - Explore data communication media (guided and unguided) and discuss channel capacity.
    - Introduce computer networks, covering network topology, types (LAN, MAN, WAN, CCAN, PAN), and network devices.
    - Provide a basic understanding of email, search engines, chatting, internet conferencing, and intranet

5. **Introduction to System Software and Operating System (15 Hours)**
    - Define the operating system, its functions, and the need for OS.
    - Classify OS based on CUI & GUI and Single or Multi-User systems.
    - Introduce concepts of Multi Programming, Multi-Tasking, and Multi Processing.
    - Explain the booting process and provide a basic understanding of Assembler, Loader, Linker, and Interpreter

UG/I/COMP/4/MI-1(P): Office Automation **Credits 01**

1. **M.S Word**
   - Introduction to Microsoft Word: Comprehensive overview of the Microsoft Word interface, encompassing document creation and seamless navigation.
   - Document Formatting: Mastery of text formatting, paragraph structuring, and meticulous document layout customization.
   - Utilizing Styles and Themes: Proficient application and tailored customization of styles, harnessing document themes for harmonized visual presentation.
   - Effective Document Management: Seamless integration of headers, footers, page numbering, tables, and graphics for comprehensive document management.

2. **Microsoft PowerPoint**
   - Creating Dynamic Presentations: Insightful introduction to the PowerPoint interface, empowering students to craft engaging slides and incorporate compelling content.
   - Customizing Slide Formats: Proficient application of slide layouts, themes, and meticulous customization of slide backgrounds to achieve visual impact.
   - Integrating Multimedia Elements: Seamless insertion of images, audio, and video files to enhance multimedia-rich presentations.
   - Elevating Presentation Delivery: Skillful application of animations and transitions to elevate presentation delivery and captivate the audience.

3. **Microsoft Excel**
   - Essential Spreadsheet Fundamentals: Comprehensive introduction to the Excel interface, encompassing efficient data entry and fundamental formula application.
   - Harnessing Data Analysis Tools: Mastery of sorting, filtering, and conditional formatting tools to facilitate data analysis and interpretation.
   - Visualizing Data with Charts and Graphs: Proficient creation and tailored customization of charts and graphs to effectively visualize data trends.

- Exploring Advanced Functions: Insightful introduction to advanced functions such as VLOOKUP, SUMIF, and COUNTIF to unlock powerful data manipulation capabilities.

4. **Internet and Email:**
    - Demonstrate how to sign up and sign in to Gmail and navigate through the Gmail interface.
    - Compose an email to someone and include the subject, recipient's email address, and a message describing the event details.
    - Create a meeting in Google Meet and participate in the meeting by interacting with other participants.
    - Join a Google Classroom using the provided class code and access the course materials shared by the instructor.
    - Submit an assignment through Google Classroom and verify the submission.
    - Create a new document in Google Docs and share the document with your classmate and make edits to the document simultaneously with your classmate and observe real-time changes.
    - Create a Google Form for conducting a survey about favourite movie genres and include at least five questions related to movie preferences, share the form link with your classmates and collect responses.

Semester II

MAJOR 2

UG/II/COMP/4/MJ-2: Digital Logic Design — Credits 04 [Full Marks: 75]

Course Objective:

- Acquire a deep understanding of positional number systems, including Binary, Octal, Hexadecimal, and Decimal, and learn how to perform conversions between them.
- Grasp the techniques for representing signed numbers, including signed magnitude, one's complement, and 2's complement, and understand their significance in digital circuits.
- Explore various binary codes such as BCD, excess-3, Gray code, ASCII, EBCDIC, and Parity bits, and comprehend their applications in data representation and communication.
- Develop proficiency in performing binary arithmetic operations, including addition, subtraction, multiplication, and division, using both manual methods and digital circuitry.
- Define Switching Algebra, comprehend its properties, and understand Huntington's Postulates, laying the foundation for logical design in digital circuits.
- Utilize basic logic gates (OR, AND, NOT) and universal logic gates (NAND & NOR) to implement logical operations and construct complex digital circuits.
- Perform logical operations including logical sum (OR), logical product (AND), complementation (NOT), Anti-coincidence (EX-OR), and Coincidence (EX-NOR), enabling the manipulation of digital signals.
- Master the techniques for simplifying Boolean expressions using algebraic methods and Karnaugh map methods, facilitating the optimization of digital circuits.
- Design a variety of digital circuits, including Half Adder, Full Adder, Half Subtractor, Full Subtractor, Multiplexers, Encoders, Demultiplexers, Decoders, Seven Segment Display, BCD adder/subtractor, comparators, parity generators, code converters, and priority encoders, to solve practical problems in digital systems.
- Understand the principles of flip-flops including latch, RS, D, JK, T Flip Flops, and implement circuits using these devices.
- Analyze circuits with race conditions and implement solutions using Master-Slave JK Flip Flop to mitigate timing issues.
- Design different types of registers including Serial Input Serial Output (SISO), Serial Input Parallel Output (SIPO), Parallel Input Serial Output (PISO), Parallel Input Parallel Output (PIPO), Universal Shift Registers, and understand their applications in digital systems.
- Implement both Asynchronous and Synchronous Counters, gaining insights into the principles of digital counting and sequencing.

Course Outline:
UG/II/COMP/4/MJ-2(T): Digital Logic Design (45 Hours) **Credits 03**

1. **Number Systems (10 Hours)**
    - Introduction to positional number systems: Binary, Octal, Hexadecimal, and Decimal.
    - Binary arithmetic operations: addition, subtraction, multiplication, and division.
    - Number system conversions: binary to decimal, decimal to binary, binary to octal, octal to binary, binary to hexadecimal, and hexadecimal to binary, BCD (Binary Coded Decimal).
    - Signed number representation: signed magnitude, one's complement, and 2's complement.
    - Application of number systems in digital circuits.
2. **Boolean Algebra (10 Hours)**
    - Fundamentals of Boolean algebra: Boolean expressions, Boolean operators (AND, OR, NOT), and Boolean laws.
    - Basic logic gates: AND gate, OR gate, NOT gate.
    - Universal logic gates: NAND gate, NOR gate.
    - Exclusive gates: Ex-OR, Ex-NOR.
    - Simplification of Boolean expressions using algebraic manipulation, truth tables, and Karnaugh maps.
    - Application of Boolean algebra in logic circuit design.
3. **Combinational Circuits (13 Hours)**
    - Design of basic combinatorial logic circuits: Half Adder, Full Adder, Half Subtractor, Full Subtractor.
    - Multiplexers: design, applications, and implementation.
    - Demultiplexers: design, applications, and implementation.
    - Encoders: design, applications, and implementation.
    - Decoders: design, applications, and implementation.
    - Seven Segment Display: design, applications, and implementation.
    - BCD adder/subtractor: design, applications, and implementation.
    - Digital comparators: design, applications, and implementation.
    - Parity Generator and Parity Checker: design, applications, and implementation.
    - Analysis and optimization of combinatorial circuits.
4. **Sequential Circuits (12 Hours)**

- Introduction to sequential logic circuits: latch, RS flip-flop, D flip-flop, JK flip-flop, T flip-flop.
- Edge Triggered Flip-Flops.
- Master-Slave JK Flip Flop: design, implementation, and analysis.
- Registers: Serial Input Serial Output (SISO), Serial Input Parallel Output (SIPO), Parallel Input Serial Output (PISO), Parallel Input Parallel Output (PIPO), Universal Shift Registers.
- Asynchronous Counters: design, applications, and implementation.
- Synchronous Counters: design, applications, and implementation.
- Analysis and optimization of sequential circuits.

UG/II/COMP/4/MJ-2(P): Digital Logic Design Lab **Credits: 01**

1. **Exploring Basic Logic Gates and Universal Gates:**
   i. Engage in experimental exploration of fundamental logic gates: AND, OR, NOT.
   ii. Implementation of different functions using Basic and Universal Logic gates, SOP, POS
   iii. Implementation of Basic gates using NAND and NOR gates.
   iv. Utilize sophisticated electronic components or advanced simulation software to execute and analyze basic logic gate circuits.
2. **Realization of Karnaugh Maps (K-Maps):**
   i. Implement and design the circuit for the expressions derived from F(A, B, C, D) = Σm(0, 1, 3, 5, 7, 8, 9, 11, 13, 15) using K-Maps.
   ii. Implement and design the circuit for the expressions derived from F(A, B, C, D) = Σm(4,5,6, 7, 11, 12) using K-Maps.
3. **Combinational Circuits:**
   i. Design and implement a Half Adder circuit.
   ii. Design and implement a Full Adder circuit.
   iii. Design and implement a Half Subtractor circuit.
   iv. Design and implement a Full Subtractor circuit.
   v. Program to design and implement a Multiplexer.
   vi. Program to design and implement a Demultiplexer.
   vii. Program to design and implement an Encoder.
   viii. Program to design and implement a Decoder.
   ix. Program to design and implement a Seven Segment Display driver.
   x. Program to design and implement a BCD Adder/Subtractor circuit.

xi. Program to design and implement a Digital Comparator.
xii. Program to design and implement a Parity Generator.
xiii. Program to design and implement a Parity Checker.

4. **Sequential Circuits**:
   i. Design and implement an RS flip-flop.
   ii. Design and implement a D flip-flop.
   iii. Design and implement a JK flip-flop.
   iv. Design and implement a T flip-flop.
   v. Program to design and implement a master-slave JK flip-flop.
   vi. Program to implement a Serial Input Serial Output (SISO) register.
   vii. Program to implement a Serial Input Parallel Output (SIPO) register.
   viii. Program to implement a Parallel Input Serial Output (PISO) register.
   ix. Program to implement a Parallel Input Parallel Output (PIPO) register.
   x. Program to design and implement an Asynchronous Counter.
   xi. Program to design and implement a Synchronous Counter.

Reference Books

For Digital Logic Design

1. **Malvino and Leach, “Digital Principles and Applications”.**
2. **R.P. Jain,“Modern Digital Electronics” .**
3. **S.Salivahanan, S.Arivazhagan ,“Digital Circuits & Design” — Vikas Publishing House Pvt Ltd.**
4. **M.Mano, “Digital logic & Computer Design“,Prentice Hall of India.**
5. **A. Annand Kumar ,“Fundamental of Digital Circuits”.**
6. **David Harris and Sarah Harris ,"Digital Design and Computer Architecture".**
7. **Guy Even and Moti Medina ,"Digital Logic Design: A Rigorous Approach" by**
8. **Ronald J. Tocci, Neal S. Widmer, and Greg Moss ,"Digital Systems: Principles and Applications".**

MINOR 2

UG/II/COMP/4/MI-2: Introduction to Programming in C Credits 04 [Full Marks: 75]

Course Objectives:

- Cultivate a profound understanding of programming languages, focusing on 'C,' and encompassing fundamental programming concepts.
- Illuminate key aspects such as Loops, Data reading, stepwise refinement, Functions, Control structures, and Arrays, fostering a holistic understanding of programming fundamentals.
- Develop the capability to analyze real-world problems and adeptly devise solutions through programming in 'C.'
- Prioritize problem-solving skills, emphasizing the creation of effective algorithms applicable in 'C.'
- Equip students with the ability to formulate efficient algorithms for diverse problem scenarios in the 'C' programming language.
- Familiarize students with the various constructs of programming languages, encompassing conditional statements, iteration, and recursion in 'C.'
- Enable students to implement devised algorithms effectively using the "C" language.
- Provide hands-on experience in utilizing fundamental data structures like arrays, stacks, and linked lists for problem-solving in 'C.'
- Empower students with the skills to manage file operations adeptly within the context of "C" programming.

UG/II/COMP/4/MI-2(T): Introduction to Programming in C Credits: 03

Course Outline:

Programming in C (45 Hours)

1. **Introduction to Programming (4 Hours)**
    - Introduction to various programming styles like procedural, object-oriented, and functional.
    - Understanding the unique features and practical applications of each style.
    - Examining the role of programming in addressing real-world challenges.
    - Introduction to Symbolic Constant, Pre-Processor Directives and Header Files.
    - Understanding the process of analyzing and deconstructing problems, including identifying inputs and output.
    - Basics of thinking algorithmically and expressing ideas using pseudocode and Flowchart.
    - Introduction to essential algorithms such as searching, sorting, and recursion.
2. **Conditional Statements and Loops (8 Hours)**
    - Overview of the syntax and structure of the C programming language.
    - Understanding different types of variables, data types, and operators in C.

- Learning basic input and output operations in C.
- Exploring control flow statements like if-else, nested if-else, switch-case, break, continue and goto.
- Learning about various loop structures like for loop, while loop, and do-while loop.
- Applying conditional statements and loops in practical scenarios with examples.

3. **Arrays and Strings (10 Hours)**
    - Introduction to arrays and their manipulation techniques in C.
    - Exploring string handling functions like strcpy, strcat, strlen, etc.
    - Practical exercises focusing on array manipulation and string handling.
4. **Functions and Modular Programming (8 Hours)**
    - Basics of functions including declaration, definition, and invocation.
    - Understanding parameter passing mechanisms such as by value and call by references, Recursive functions, Macro function.
    - Exploring the principles and advantages of modular programming.
    - Implementing modular programs through practice sessions.
5. **Storage Classes and Scope (4 Hours)**
    - Exploring variable scope concepts including local, global, and static.
    - Overview of storage classes such as auto, extern, static, and register.

6. **Structures and Union (6 Hours)**
    - Basics of structure declaration and initialization.
    - Understanding pointer fundamentals including declaration, dereferencing, and arithmetic operations.
    - Exploring the applications of structures and pointers in programming.

7. **Pointers (5 Hours)**
    - Discussion on memory address operators, explanation of declaring pointer types.
    - Assignment and initialization of pointers, performing arithmetic operations using pointers.
    - Functions that utilize pointers.
    - Exploring the connection between arrays and pointers
    - Handling pointers within structures.

UG/II/COMP/4/MI-2(P): Programming in C Lab **Credits: 01**

1. Write a C program to check whether a given number is even or odd.
2. Write a C program to calculate the area of a rectangle.
3. Write a C program to calculate the sum of first n even numbers.
4. Write a C program to read a number and find factorial.
5. Write a C program to find largest number among three numbers.
6. Write a C program to check whether a given number is Amstrong or not.
7. Write a program to check whether a given number is prime or not.
8. Write a program to generate the Fibonacci series up to a given number.
9. Write a program to check whether a given string is a palindrome or not.
10. Write a C program to implement a simple calculator that performs addition, subtraction, multiplication, and division.
11. Implement a C program to find the largest element in an array.
12. Write a C program to concatenate two strings.
13. Write a C program to swap two numbers.
14. Write a C program to perform linear search to find the position of a given element in an array.
15. Write a C program to sort an array of integers in ascending order.

Reference Books:

For C Programming Language

1. **Byron S Gottfried "Programming with C", Fourth edition, Tata McGrawhill.**
2. **E. Balagurusamy, "Programming with ANSI-C", Eight Edition,2008, Tata McGraw Hill.**
3. **Kanetkar Y, "Let us C", BPB Publications, 2017.**
4. **Brian W. Kernighan & Dennis Ritchie, "The C Programming Language", Second Edition, Pearson.**

SEC 2

UG/II/COMP/4/SE-2 P: Problem Solving Using Python — Credits: 03

Course Objective:

- Introduce fundamental features of Python programming, focusing on industry standards.
- Enable students to apply advanced Python programming features to solve real-world problems.
- Instill a solid understanding of Python programming concepts, libraries.

Course Outline:

- Develop the capability to create applications in various fields using Python.
- Explore computer systems and the Python programming language.
- Emphasize computational thinking and cover Python data types, expressions, operators, variables, assignments, strings, lists, and the Python standard library.
- Dive into imperative programming, covering Python modules and built-in functions like print() and eval().
- Develop user-defined functions and assignments, emphasizing parameter passing.
- Focus on text data, files, and exceptions, covering strings, formatted output, files, errors, and exceptions.
- Introduce execution control structures, decision control, and the IF statement.
- Cover For LOOP and iteration patterns, including two-dimensional lists, while loop, additional loop patterns, and iteration control statements.
- String, List, Tuples, Dictionaries, Sets.

Python Practical:

1. Write a menu driven program to convert the given temperature from Fahrenheit to Celsius and vice versa depending upon users' choice.
2. Check if a number is prime or not.
3.
4. Write a Program to calculate total marks, percentage and grade of a student. Marks obtained in each of the three subjects are to be input by the user. Assign grades according to the following criteria:

   Grade A: Percentage >=80

   Grade B: Percentage>=70 and <80

Grade C: Percentage>=60 and <70
Grade D: Percentage>=40 and <60
Grade E: Percentage<40

5. Write a menu-driven program, using user-defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.
6. Write a Program to display the first n terms of Fibonacci series.
7. Write a Program to find factorial of the given number.
8. Write a Program to find sum of the following series for n terms: 1 – 2/2! + 3/3! - - - - - n/n!
9. Write a Program to calculate the sum and product of two compatible matrices.
10. Create two matrices and perform matrix multiplication using NumPy.
11. Write a program to find the square root of a given number.

Reference books:

1. **"Python Programming: A Modular Approach with Graphics, Database, Mobile, and Web Applications" by Sheetal Taneja & Naveen Kumar, Pearson, 2017.**
2. **"Python: The Complete Reference" by Martin C. Brown, Osborne/McHraw Hill, 2001.**
3. **"Core Python Programming" by Wesley J. Chun, Pearson Education, Second Edition, 2007.**
4. **Introduction to Computing Using Python: An Application Development Focus" by Ljubomir Perkovic, John Wiley & Sons, 2012.**