|  |  |  |  |
| --- | --- | --- | --- |
| Institute/ School Name | School of Engineering and Technology | | |
| Department Name | Department of Computer Science & Engineering | | |
| Program Name | Bachelor of Engineering (Computer Science &amp; Engineering): B.E (CSE) | | |
| Course Code |  | Course Name | Computer Programming II |
| L-T-P (Per Week) | 4-0-2 | Course Credits | 05 |
| Academic Year | 2023-24 | Semester/Batch | 2nd /2023-2027 |
| Course Coordinator |  | | |

1. **Course Outline:**

Recap of Function and Pointer, Searching-Linear Search and Binary Search, Sorting- Bubble, Insertion, Selection, Quick Sort, 2D Array, String and functions, String and pointers, Recursion- Direct and Indirect, Tail and Non-Tail Recursion, Dynamic Memory Allocation, Structure and Union, Stack, Queue, Bit Manipulation

1. **Programme Outcomes (POs):**

|  |  |
| --- | --- |
| At the end of the programme, students will be able to: | |
| PO 1 | **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| PO 2 | **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| PO 3 | **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| PO 4 | **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| PO 5 | **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations. |
| PO 6 | **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| PO 7 | **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| PO 8 | **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| PO 9 | **Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| PO10 | **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| PO11 | **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| PO12 | **Life-long learning**: Recognize the need for and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

1. **Course Learning Outcomes (CLO):**

Student should be able to

**CLO1:** Understand the concept of Functions and should be able to divide the complex real-world problem into

subproblems and provide the solutions for the subproblem in form of functions

**CLO2:** Understand and apply pointers.

**CLO3:** Work with array, strings, stack and queues

**CLO4:** Understand and apply searching and sorting algorithms

**CLO5:** Use recursion, dynamic memory allocation technique

1. **CLO-PO Mapping Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Learning Outcomes** | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
| CLO1 |  | H | H |  | L |  |  |  |  |  | M | H |
| CLO2 |  |  | H | L |  |  |  |  |  |  |  | M |
| CLO3 |  |  | H | L | M |  |  |  |  |  |  | M |
| CLO4 |  |  | H | L | M |  |  |  |  |  |  | M |
| CLO5 |  |  | H | L | M |  |  |  |  |  |  | M |

1. **ERISE Grid Mapping:**

|  |  |
| --- | --- |
| **Feature Enablement** | **Level(1-5, 5 being highest)** |
| Entrepreneurship | 1 |
| Research/Innovation | 1 |
| Skills | 5 |
| Employability | 4 |

**Recommended Books (Reference Books/Text Books):**

**B01:** Programming in ANSI C by E. Balagurusamy, McGraw Hill Education, Eighth Edition

**B02:** Head First C by David Griffiths and Dawn Griffiths, Shroff/O’Reilly

**B03:** C: The Complete Reference by Herbert Schildt, McGraw Hill Education, Fourth Edition

1. **Other readings and relevant websites:**

|  |  |
| --- | --- |
|  | **Link of Journals, Magazines, Websites and Research Papers** |
| **R1** | https://ocw.mit.edu/courses/6-087-practical-programming-in-c-january-iap-2010/pages/lecture-notes/ |
|  |  |
|  | **Link of Audio-Video resources** |
| **V1** | https://www.youtube.com/playlist?list=PLhQjrBD2T382eX9-tF75Wa4lmlC7sxNDH |

\* Resources uploaded on ERP system is accessible to all the students registered for the course.

1. **Recommended Tools and Platforms:**

* Visual Studio Code
* Code Quotient (online platform-https://codequotient.com)

1. **Course Plan:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lecture Number** | **Topics** | **Weightage in ETE (%)** | **Instructional Resources** |
| 1-4 | Recap of Programs to cover Basic C constructs like, variables, data types, keywords, decision statements and loops | 30% | B01, B02 |
| 5-6 | Recap on Functions and pointers |
| 7-8 | Null Pointers, Void Pointers and Pointers with const keyword |
| 9-10 | Searching: Linear Search, Application of Linear Search |
| 11-12 | Searching: Binary Search and its application |
| 13-14 | Introduction to Sorting and Its Uses, Bubble sort |
| 15-16 | Insertion and Selection Sort |
| 17-18 | Quick Sort |
| 19-20 | Recap of 2D Arrays |
| 21-22 | Pointers and 2D Arrays |
| 23-24 | String: library functions, Strings and functions | 10% | B01, B02 |
| 25-26 | String and pointers |
| 27-28 | Recursion: Recursion Example, Order of Execution | 15% | B01, B02 |
| 29-30 | Direct and Indirect Recursion, Stack overflow |
| 31-32 | Tail and Non-Tail Recursion |
| 33-34 | Practice Problem:   * Find nth Fibonacci number using tail recursion * Write a recursive function to find the sum of n natural numbers where n >= 1 |
| 35-36 | DMA (Dynamic Memory Allocation), Memory model of C Program, Concept of Dynamic Memory allocation | 15% | B01, B02 |
| 37-38 | DMA: malloc(), calloc() |
| 39-40 | DMA: realloc() and free(), Memory leak in C |
| 41-42 | Practice Problem: Find largest number out of set of numbers. Numbers are stored using dynamic memory allocation |
| 43-44 | Introduction to Structures, typedef | 15% | B01, B02 |
| 45-46 | Structure Pointers and Nested Structures |
| 47-48 | Structure and Functions, Structure Assignment |
| 49-50 | Array of Structures |
| 51-52 | Union, Union vs Structure, Enumeration |
| 53-54 | Stack: Implementing stack using arrays, reversing string using stack | 15% | B01, B02 |
| 55-56 | Queue: Implementing queue using arrays |
| 57-58 | Bit Manipulation: Decimal to Binary Conversion, AND, OR, XOR Operator |
| 59-60 | Bit Manipulation: NOT, Left shift and Right shift operators |

1. **Industry Interventions:**

* Industry Curated Module: <https://codequotient.com>
* Expert Session on File handling

1. **Action plan for different types of learners**

|  |  |  |
| --- | --- | --- |
| **Slow Learners** | **Average Learners** | **Advanced Learners** |
| Remedial Classes | Workshops/Practice Assignment | Coding/Interclass Competition |

1. **Evaluation Scheme & Components:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation Component** | **Type of Component** | **No. of Assessments** | **Weightage of Component** | **Mode of Assessment** |
| Internal Component 1 | Formative Assessments (FAs) | 02\* | 20% | Online Platform |
| Internal Component 2 | Sessional Tests (STs) | 03\*\* | 30% | Online Platform |
| External Component | End Term Examination (ETE) | 01 | 50% | Online Platform |
| **Total** | | **100%** | | |

\* There will be 02 FAs, both are mandatory, and the ERP system automatically picks the average of both for the

calculation of final marks.

\*\* Out of 03 STs, the ERP system automatically picks the best 02 ST marks to evaluate final marks.

1. **Details of Evaluation Components:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Evaluation Component** | **Description** | **Syllabus Covered (%)** | **Timeline of Examination** | **Weightage (%)** |
| Internal Component 1 | FA1 | Upto 25% (Lecture 1- 15) | Week 4 | 20% |
| FA2 | 26% - 50% (Lecture 16 – 30) | Week 9 |
| Internal Component 1 | ST 01 | Upto 40% (Lectures 1-24) | Week 4 | 30% |
| ST 02 | 40% - 80% (Lectures 25-48) | Week 8 |
| ST 03 | 100% (Lectures 1-60) | Week 11 |
| External Component | End Term Examination\* | 100% | As Notified by the Exam Cell | 50% |
| **Total** | | | | 100% |

\* Minimum 75% attendance is required to become eligible for appearing in the End Semester Examination

1. **Format of Evaluation Components:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Type of Assessment** | **Total Marks** | **Module Coverage** | **1 Mark MCQ** | **2 Marks MCQ** | **5 Marks Coding** | **10 Marks Coding** |
| \*Formative  Assessments | 20 | 10\* | 10\* | - | - | - |
| Sessional Tests | 40 | - | 10 | 5 | 2 | 1 |
| End Term Examination | 60 | - | 10 | 10 | 4 | 1 |

\*Formative assessment will carry weightage: 10 marks for code quotient module completion, and 10 marks for

MCQs.

1. **This Document is approved by:**

|  |  |  |
| --- | --- | --- |
| **Designation** | **Name** | **Signature** |
| **Prepared by Course Coordinator** |  |  |
| **Verified by Program Incharge/HoD** | Dr. Madhu Gupta |  |
| **Approved by Pro VC** | Prof. (Dr.) Meenu Khurana |  |
| **Date** |  | | |

**\*Incase of revision in earlier document:**

|  |  |  |  |
| --- | --- | --- | --- |
| Date of Creation of earlier document |  | Percentage of Revision |  |

Annexure-1

|  |  |
| --- | --- |
| S. No. | Practical List |
| 1. | **Write a C program to find all roots of a quadratic equation using if else.**  Quadratic equation  Quadratic equation  A quadratic equation can have either one or two distinct real or complex roots depending upon nature of discriminant of the equation. Where discriminant of the quadratic equation is given by  Discriminant of a quadratic equation  Depending upon the nature of the discriminant, formula for finding roots is be given as.   * Case 1: If **discriminant is positive**. Then there are two real distinct roots given by. Quadratic equation formula root 1 * Case 2: If **discriminant is zero** then, it has exactly one real root given by. Quadratic equation formula root 2 * Case 3: If **discriminant is negative** then, it has two distinct complex roots given by. Quadratic equation formula root 3   **Example**  **Input**  Input a: 8  Input b: -4  Input c: -2  **Output**  Root1: 0.80  Root2: -0.30  **Write a C program to create menu driven calculator that performs basic arithmetic operations (add, subtract, multiply and divide) using switch case**  **Example**  **Input:** 5.2 - 3  **Output:** 2.2 |
| 2. | **Write a C program to input a number and calculate products of digits using loop.**  **Example:**  **Input** 1234  **Output**  24  -------------------------------------------------------------------------------------------------------  **Write a C program to input two numbers from user and find LCM (Lowest Common Multiple) using loop.**  **Example:**  **Input**  Input number1: 12  Input number2: 30  **Output**  LCM: 60 |
| 3. | **Write a C program to input binary number from user and find twos complement of the binary number**  Twos complement of binary value  **Example**  **Input binary number:** 01101110  **Output:**  **Twos complement:** 10010010  **Write a C program to input decimal number from user and convert to binary number system**  **Example**  **Input**  Input decimal number: 112  **Output**  Binary number: 0111000 |
| 4. | **Write a function in C programming to find all prime numbers within a range from lower limit to upper limit given by user (using function.)**  **Example**  **Input**  Input lower limit: 10  Input upper limit: 50  **Output**  Prime numbers between 10-50 are: 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47  **Write a function to print all Armstrong numbers between given interval in C programming**  **Example**  **Input**  Input lower limit: 1  Input upper limit: 1000  **Output**  Armstrong numbers between 1 to 1000 are: 1, 153, 370, 371, 407 |
| 5. | **Write a C program to copy one array elements to another array using pointers.**  **Example**  **Input**  Input array1 elements: 10 -1 100 90 87 0 15 10 20 30  **Output**  Array1: 10 -1 100 90 87 0 15 10 20 30  Array2: 10 -1 100 90 87 0 15 10 20 30  **Write a C program to return multiple value from function. (pass array to function)**  **Hint:** If you want to return multiple similar type values from a single function. Then returning an array as a set of values is best suited. |
| 6. | **Write a C program to input elements in array and search whether an element exists in array or not. (Linear Search)**  **Example**  **Input**  Input size of array: 10  Input elements in array: 10, 12, 20, 25, 13, 10, 9, 40, 60, 5  **Output**  Element to search is: 25  Element found at index 3  **Write a C program to implement a binary search algorithm**  **Example**  **Input**  Input size of array: 10  Input elements in array: 10, 12, 20, 25, 33, 47, 98, 400, 678, 1005  **Output**  Element to search is: 25  Element found at index 3 |
| 7. | **Write a C program to implement insertion sort algorithm**  **Example**  **Input**  Input size of array: 10  Input elements in array: 10, 12, 20, 25, 13, 10, 9, 40, 60, 5  **Output**  Sorted array: 5, 9, 10, 10, 12, 13, 20, 25, 40, 60  **Write a C program to implement bubble sort algorithm**  **Example**  **Input**  Input size of array: 10  Input elements in array: 10, 12, 20, 25, 13, 10, 9, 40, 60, 5  **Output**  Sorted array: 5, 9, 10, 10, 12, 13, 20, 25, 40, 60 |
| 8. | **Write a C program to implement Quick sort algorithm**  **Example**  **Input**  Input size of array: 10  Input elements in array: 10, 12, 20, 25, 13, 10, 9, 40, 60, 5  **Output**  Sorted array: 5, 9, 10, 10, 12, 13, 20, 25, 40, 60 |
| 9. | **Write a C program to read elements in a matrix and find the sum of elements of each row and columns of matrix.**  **Example**  **Input**  Input elements in array:  1 2 3  4 5 6  7 8 9  **Output**  Sum of row 1 = 6  Sum of row 2 = 15  ...  ...  Sum of column 3 = 18  **Write a C program to read elements in a matrix and check whether matrix is Sparse matrix or not.**  **Example**  **Input**  Input elements in matrix:  1 0 3  0 0 4  6 0 0  **Output**  The given matrix is Sparse matrix |
| 10. | **Write a C program to multiply two matrix using pointers.**  **Example**  **Input**  Input elements of matrix1:  10 20 30  40 50 60  70 80 90  Input elements of matrix2:  1 2 3  4 5 6  7 8 9  **Output**  Product of matrices is:  300 360 420  660 810 960  1020 1260 1500 |
| 11. | **Write a function in C that compares two strings lexicographically and returns 0 if both strings are equal, negative if first string is smaller otherwise returns positive (without using library function)**  **Example**  **Input**  Input string1: Learn at Codeforwin.  Input string2: Learn at Codeforwin.  **Output**  Both strings are lexicographically equal. |
| 12. | **Write a recursive function in C to find GCD (HCF) of two numbers**  **Example**  **Input**  Input first number: 10  Input second number: 15  **Output**  HCF of 10 and 15 = 5 |
| 13. | **Write a C program to find sum of array elements using recursion**  **Example**  **Input**  Input size of array: 10  Input array elements: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  **Output**  Sum of array: 55 |
| 14. | **Write a C program to find out the maximum and minimum from an array (stored using dynamic memory allocation).**  **Example**  **Input**  Input size of array: 10  Input array elements: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10  **Output**  Maximum: 10  Minimum: 1 |
| 15. | **Write a C program to declare and initialize the nested structure as per details given below:**  Create a structure address with fields House\_number, Street\_number, State and pin\_code  Create a nested structure Employee having fields Empid, EmpName, Organization, address (type of structure defined above)  **Note:** data types and dummy data can be assumed as per context of field |
| 16. | **Write a C program to store and display the record of 5 students as per the details given below:**  Each student following details need to be captured:  Name, University\_id, Program, Batch, Semester  **Note:** data types and dummy data can be assumed as per context of field |
| 17. | **Write a C program to extract individual bytes from an unsigned int [using union]**  **Example**  **Input**  Unsigned Int: 2864434397  **Output**  Individual bytes: AA BB CC DD  **Write a C program to create a union with two members of type int and float respectively. Calculate the size of union** |
| 18. | **Write a C program to print the value of weekdays using enumerated data type**  **Example**  **Input**  Sunday  **Output**  **6**  **Write a C program to create and use flags using enums as per details given below**  Use enum in C for flags by keeping the values of integral constants a power of 2. This will allow us to choose and combine two or more flags without overlapping with the help of the Bitwise OR (|) operator  Create three flags CROP with value 1, ROTATE with value 2 and SAVE with value 4  **Example:**  Operation performed ROTATE | SAVE  Output: 6 |
| 19. | **Write a C program to stack using arrays and also code functions for push and pop operations** |
| 20. | **Write a C program to queue using arrays and also code functions for enqueue and dequeue operations** |
| 21. | **Write a C program to input any number from user and check whether the Least Significant Bit (LSB) of the given number is set (1) or not (0)**  **Example**  **Input**  Input number: 11  **Output**  Least Significant Bit of 11 is set (1).  **Write a C program to input any number from user and check whether Most Significant Bit (MSB) of given number is set (1) or not (0).**  **Example**  **Input**  Input number: -1  **Output**  Most Significant Bit (MSB) of -1 is set (1). |
| 22. | **Write a C program to input a number from user and flip all bits of the given number (in binary representation) using bitwise operator**  **Example**  **Input**  Input any number: 22  **Output**  Number after bits are flipped: -23 (in decimal)  **Write a C program to input any decimal number from user and convert it to binary number system using bitwise operator**  **Example**  **Input**  Input any number: 22  **Output**  Binary number: 00000000000000000000000000010110 |