Name of the Course: Computer Science Practical 2

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Computer Science Practical Course covering Design and Analysis of Algorithms and Object-Oriented Programming (OOP) using C++ is a comprehensive exploration into fundamental computer science concepts and practical programming skills. It integrates the study of algorithmic design with hands-on application using the C++ programming language. **Relevance:** In the dynamic field of computer science, the integration of algorithmic design and object-oriented programming is highly relevant. This course equips students with essential skills to solve complex problems, design efficient algorithms, and implement practical solutions using the OOP paradigm in C++. **Usefulness:** The course is invaluable for developing a strong foundation in algorithmic thinking and software design. Students learn to analyze algorithm efficiency, apply OOP principles for code modularity, and create robust software solutions, enhancing their overall programming proficiency. **Application:** The concepts acquired in this practical course find direct application in real-world scenarios. Students engage in hands-on projects where they design and implement algorithms, analyze their performance, and develop software applications using object-oriented principles in C++. **Interest:** The practical nature of the course often captivates students. Through project-based learning, participants apply algorithmic strategies, design class hierarchies, and implement solutions in C++, fostering a deep interest in problem-solving and software development. **Connection with Other Courses:** This practical course establishes a strong connection with other computer science courses. It lays the groundwork for advanced studies in algorithmic complexity, data structures, software engineering, and advanced topics in object-oriented programming, providing a well-rounded education.

		Demand in the Industry: Professionals with proficiency in algorithmic design and object-oriented programming in C++ are in high demand. Industries spanning software development, technology, and finance actively seek individuals who can apply these skills to create efficient and scalable software solutions. **Job Prospects:** Graduates from this practical course have diverse job prospects. Roles may include software engineer, algorithm developer, systems analyst, or application developer. These professionals are valued for their ability to contribute to algorithmically optimized, modular, and maintainable software.
2	**Vertical:**	Major
3	**Type:**	Practical
4	**Credits:**	2 credits (1 credit = 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	60 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives(CO):** **CO 1.** Analyze and implement algorithms for common computational problems. **CO 2.** Implement algorithms using divide and conquer strategies. **CO 3.** Apply dynamic programming techniques to solve optimization problems. **CO 4.** Implement and analyze algorithms based on greedy strategies. **CO 5.** Comprehend the principles of object-oriented programming. **CO 6.** Design and implement classes and objects in C++. **CO 7.** Implement single, multiple, and hierarchical inheritance. **CO 8.** Implement operator overloading for user-defined types. **CO 9.** Understand the impact of access specifiers on class members.	
8	**Course Outcomes (OC):** **OC 1.** Design and implement algorithms for various problem domains. **OC 2.** Evaluate and compare the time and space complexities of algorithms. **OC 3.** Apply divide and conquer strategies to solve computational problems. **OC 4.** Utilize dynamic programming techniques for optimization problems. **OC 5.** Implement and analyze algorithms based on greedy strategies. **OC 6.** Design and implement classes and objects in C++. **OC 7.** Apply inheritance and polymorphism concepts in program development. **OC 8.** Implement operator overloading for enhanced class functionality. **OC 9.** Utilize advanced features like friend functions, inline functions, and this pointer. **OC 10.** Understand the impact of scope specifiers on class members.	

9	**Modules:-** **Module 1 (30 hours):**
	Design & Analysis of Algorithms – Practical **Array Operations:** Implement programs for 1-d arrays, Implement programs for 2-d arrays. **List-Based Stack Operations:** Create a list-based stack and perform stack operations. **Linear and Binary Search:** Implement linear and binary search algorithms on a list. **Sorting Algorithms:** Implement sorting algorithms (e.g., bubble, selection, insertion). **Nth Max/Min Element:** Implement algorithms to find Nth Max/Min element in a list. **String Pattern Matching:** Implement algorithms to find a pattern in a given string. **Recursion:** Implement recursive algorithms (e.g., factorial, Fibonacci, Tower of Hanoi). **Greedy Algorithm:** Solve problems like file merging and coin change using the Greedy Algorithm. **Divide and Conquer:** Implement algorithms like merge sort and Strassen's Matrix Multiplication. **Dynamic Programming:** Implement algorithms for Fibonacci series and Longest Common Subsequence using dynamic programming.
	Module 2 (30 hours):
	OOPs using C++ – Practical **Introduction to Classes:** Create a simple class with data members and member functions. Demonstrate the use of class instances to access data and invoke member functions. **Branching and Looping with Classes:** Implement programs utilizing branching and looping statements within class methods. **Arrays and Classes:**

	Develop a program that employs one and two-dimensional arrays within a class. Illustrate how classes can handle array-based data structures. **Scope Resolution Operator:** Use the scope resolution operator to declare variables at different scope levels. Display and compare the values of variables with different scopes. **Constructors and Destructors:** Implement programs showcasing various types of constructors and destructors. Explore default, parameterized, copy constructors, and destructor functionalities. **Access Specifiers:** Demonstrate the use of public, protected, and private scope specifiers within a class. Understand the impact of different access specifiers on class members. **Inheritance:** Implement classes to demonstrate single and multilevel inheritance scenarios. Showcase how derived classes inherit properties from the base class. Develop programs illustrating multiple and hierarchical inheritance. Create programs that demonstrate the interaction between inheritance and derived class constructors. Understand the order of constructor invocation in the inheritance hierarchy. **Advanced Concepts:** Implement programs showcasing friend functions, inline functions, and the use of the this pointer within classes. **Function Overloading and Overriding:** Develop programs to demonstrate function overloading and overriding within classes. **Pointers and File Handling:** Explore the use of pointers within classes, emphasizing dynamic memory allocation. Develop programs for both text and binary file handling within a class context.
10	**Text Books** 1. Data Structure and Algorithm Using Python, Rance D. Necaise, Wiley India Edition, 2016. 2. Object Oriented Programming with C++, Balagurusamy E., 8th Edition, McGraw Hill Education India.

<table>
<tr><td>11</td><td colspan="2">Reference Books
1. Data Structures and Algorithms Made Easy, Narasimha Karumanchi, CareerMonk Publications, 2016.
2. Let Us C++ by KanetkarYashwant, Publisher: BPB Publications, 2020</td></tr>
<tr><td>12</td><td>Internal Continuous Assessment: 40%</td><td>Semester End Examination: 60%</td></tr>
<tr><td>13</td><td>The internal evaluation will be determined by the completion of practical tasks and the submission of corresponding write-ups for each session. Each practical exercise holds a maximum value of 5 marks. The total evaluation, out of 100 marks, should be scaled down to a final score of 20 marks.

Total: 20 marks</td><td>A Semester End Practical Examination of 2 hours duration for 30 marks as per the paper pattern given below.

Certified Journal is compulsory for appearing at the time of Practical Exam

Total: 30 Marks</td></tr>
<tr><td>14</td><td colspan="2">Format of Question Paper:

Total Marks: 30 Duration: 2 Hours
<table>
<tr><th>Question</th><th>Practical Question Based On</th><th>Marks</th></tr>
<tr><td>Q. 1</td><td>Module 1</td><td>12</td></tr>
<tr><td>Q. 2</td><td>Module 2</td><td>12</td></tr>
<tr><td>Q. 3</td><td>Viva</td><td>06</td></tr>
</table></td></tr>
</table>

Name of the Course: Design and Analysis of Algorithms

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Design and Analysis of Algorithms course is a fundamental exploration into the systematic study of algorithms, their design principles, and the analysis of their efficiency. It forms the backbone of computer science education, providing essential skills for solving complex computational problems. **Relevance:** In the ever-evolving landscape of computer science, the Design and Analysis of Algorithms course is highly relevant. It equips students with the intellectual tools necessary to address challenges in diverse areas, from software development to artificial intelligence. **Usefulness:** This course is instrumental in cultivating algorithmic thinking. Participants learn to devise efficient algorithms, analyze their correctness, and evaluate their performance, essential skills for creating optimized solutions in various computing applications. **Application:** The knowledge gained from this course finds application in a myriad of scenarios, from developing efficient search and sorting algorithms to optimizing resource utilization in network design and artificial intelligence. **Interest:** The course often captivates students due to its intellectual challenges and problem-solving nature. Participants engage in dissecting complex problems, devising algorithmic solutions, and analyzing their efficiency, fostering a deep appreciation for algorithmic thinking. **Connection with Other Courses:** The Design and Analysis of Algorithms course establishes vital connections with other computer science disciplines. It forms the basis for advanced courses in data structures, algorithmic complexity, and computational theory, providing a holistic understanding of computation.

		Demand in the Industry: Professionals well-versed in algorithm design and analysis are in high demand. Industries ranging from technology and finance to healthcare actively seek individuals who can develop efficient algorithms to solve complex problems and enhance system performance. **Job Prospects:** Graduates from a Design and Analysis of Algorithms course find themselves well-positioned for various roles, including software engineer, algorithm developer, data scientist, and research scientist. These professionals are valued for their ability to devise elegant and efficient solutions to computational challenges.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives(CO):** **CO 1.** To make students understand the basic principles of algorithm design **CO 2.** To give idea to students about the theoretical background of the basic data structures **CO 3.** To familiarize the students with fundamental problem-solving strategies like searching, sorting, selection, and recursion and help them to evaluate efficiencies of various algorithms. **CO 4.** To teach students the important algorithm design paradigms and how they can be used to solve various real world problems	
8	**Course Outcomes (OC):** **OC 1.** Students should be able to understand and evaluate efficiency of the programs that they write based on performance of the algorithms used. **OC 2.** Students should be able to appreciate the use of various data structures as per need **OC 3.** To select, decide and apply appropriate design principle by understanding the requirements of any real life problems.	
9	**Modules:-** **Module 1 (15 hours):**	
	Introduction to algorithms - What is algorithm, analysis of algorithm, Types of complexity, Running time analysis, How to Compare Algorithms, Rate of Growth, Types of Analysis, Asymptotic Notation, Big-O Notation, Omega-Ω Notation, Theta-Θ Notation, Asymptotic Analysis, Performance characteristics of algorithms,	

	Estimating running time / number of steps of executions on paper, Idea of Computability **Introduction to Data Structures -** What is data structure, types, Introduction to Array(1-d & 2-d), Stack and List data structures, operations on these data structures, advantages disadvantages and applications of these data structures like solving linear equations, Polynomial Representation, Infix-to-Postfix conversion. **Recursion -** What is recursion, Recursion vs Iteration, recursion applications like Factorial of a number, Fibonacci series & their comparative analysis with respect to iterative version, Tower of Hanoi problem. **Basic Sorting Techniques** - Bubble, Selection and Insertion Sort & their comparative analysis
	Module 2 (15 hours):
	Searching Techniques - Linear Search and its types, Binary Search and their comparative analysis, Selection Techniques - Selection by Sorting, Partition-based Selection Algorithm, Finding the Kth Smallest Elements in Sorted Order & their comparative analysis, String Algorithms - Pattern matching in strings, Brute Force Method & their comparative analysis **Algorithm Design Techniques** - Introduction to various types of classifications/design criteria and design techniques, Greedy Technique - Concept, Advantages & Disadvantages, Applications, Implementation using problems like - file merging problem. Divide-n-Conquer - Concept, Advantages & Disadvantages, Applications, Implementation using problems like - merge sort, Strassen's Matrix Multiplication **Dynamic Programming -** Concept, Advantages & Disadvantages, Applications, Implementation using problems like - Fibonacci series, Factorial of a number, Longest Common subsequence **Backtracking Programming -** Concept, Advantages & Disadvantages, Applications, Implementation using problems like N-Queen Problem
10	**Text Books** 1. Data Structure and Algorithm Using Python, Rance D. Necaise, Wiley India Edition, 2016. 2. Data Structures and Algorithms Made Easy, Narasimha Karumanchi, CareerMonk Publications, 2016. 3. Introduction to Algorithms, Thomas H. Cormen, 3rd Edition, PHI.
11	**Reference Books** 1. Introduction to the Design and Analysis of Algorithms, Anany Levitin, Pearson, 3rd Edition, 2011. 2. Design and Analysis of Algorithms, S. Sridhar, Oxford University Press, 2014.

<table>
<tr><td>12</td><td>Internal Continuous Assessment: 40%</td><td>Semester End Examination: 60%</td></tr>
<tr><td>13</td><td>Continuous Evaluation through:
Class Test on Module 1: 10 marks
Class Test on Module 2: 10 marks
Average of 2 Class Tests: 10 marks
Assignment on Module 1: 5 marks
Assignment on Module 2: 5 marks
Total of 2 Assignments: 10 marks
Total: 20 marks</td><td>Evaluation through:
A Semester End Theory Examination of 1 hour duration for 30 marks as per the paper pattern given below.
Total: 30 marks</td></tr>
<tr><td>14</td><td colspan="2">Format of Question Paper:

Total Marks: 30 Duration: 1 Hour
<table>
<tr><th>Question</th><th>Based On</th><th>Options</th><th>Marks</th></tr>
<tr><td>Q. 1</td><td>Module 1</td><td>Any 2 out of 4</td><td>10</td></tr>
<tr><td>Q. 2</td><td>Module 2</td><td>Any 2 out of 4</td><td>10</td></tr>
<tr><td>Q. 3</td><td>Module 1 & 2</td><td>Any 2 out of 4</td><td>10</td></tr>
</table></td></tr>
</table>

Name of the Course: Introduction to OOP using C++

Sr. No.	Heading	Particulars
1	**Description the course:**	**Introduction:** The Introduction to Object-Oriented Programming (OOP) using C++ course is a foundational exploration into the principles of object-oriented programming, using the C++ programming language. This course serves as a gateway for students to understand and apply key concepts in software design and development. **Relevance:** In the contemporary software development landscape, understanding OOP principles is crucial. The C++ language, with its strong support for object-oriented features, is widely used in building robust and efficient software systems. This course is, therefore, highly relevant to the needs of modern programming. **Usefulness:** The course is instrumental in imparting essential programming paradigms such as encapsulation, inheritance, and polymorphism. Participants gain valuable skills in designing modular and reusable code, contributing to the creation of scalable and maintainable software solutions. **Application:** The concepts learned in this course find direct application in software development. Participants learn to structure code using classes and objects, facilitating the creation of efficient and well-organized programs. **Interest:** The course often captivates students due to its practical and creative aspects. Through hands-on projects, participants engage in designing and implementing solutions using OOP principles, fostering a deep interest in software design and development. **Connection with Other Courses:** This course establishes strong connections with other programming and software engineering courses. It lays the groundwork for advanced studies in software architecture, design patterns, and application development, providing a seamless transition to more

		complex programming concepts. **Demand in the Industry:** Professionals with a solid understanding of OOP using C++ are in high demand. Industries ranging from software development to embedded systems actively seek individuals who can leverage OOP principles to create efficient, modular, and maintainable code. **Job Prospects:** Students completing this course may find diverse job prospects. Roles may include software developer, systems analyst, application architect, and embedded systems engineer. These professionals are valued for their ability to contribute to the creation of robust and scalable software solutions.
2	**Vertical:**	Major
3	**Type:**	Theory
4	**Credits:**	2 credits (1 credit = 15 Hours for Theory or 30 Hours of Practical work in a semester)
5	**Hours Allotted:**	30 Hours
6	**Marks Allotted:**	50 Marks
7	**Course Objectives(CO):** **CO 1.** To make learner understand the concepts of OOP **CO 2.** To make learner understand the design of OOP through UML **CO 3.** To make learner familiar with the syntax of C++ **CO 4.** To make learner Analyze and implement concepts of OOP **CO 5.** To make learner create programs relating to OOP concepts	
8	**Course Outcomes (OC):** **OC 1.** The learner will be able to understand, remember, demonstrate, explain and describe concept of OOP **OC 2.** The learner will be able to design UML based diagrams **OC 3.** The learner will be able to illustrate the different types of control statements in C++ **OC 4.** The learner will be able to analyze and implement concept of OOP **OC 5.** The learner will be able to write and create programs relating to OOP concepts	
9	**Modules:-** **Module 1 (15 hours):**	
	Introduction to Programming Concepts: Object oriented programming paradigm, basic concepts of object oriented programming, benefits of object oriented programming, object oriented languages, applications of object oriented programming.Tokens-keywords, identifiers, constants-integer, real, character and string constants, backslash constants, features of C++ and its basic structure, simple	

	C++ program without class, compiling and running C++ program. **Data Types, Data Input Output and Operators:** Basic data types,variables, rules for naming variables, programming constants, the type cast operator, implicit and explicit type casting, cout and cin statements,operators, precedence of operators. **Decision Making, Loops, Arrays and Strings:** Conditional statements-if,if...else, switch loops- while, do...while, for, types of arrays and string and string manipulations **Unified Modeling Language (UML):** Introduction to UML & class diagrams. **Classes, Abstraction & Encapsulation:** Classes and objects, Dot Operator, data members, member functions, passing data to functions, scope and visibility of variables in function. **Constructors and Destructors:** Default constructor, parameterized constructor, copy constructor, private constructor, destructors. **Working with objects:** Accessor - mutator methods, static data and static function, access specifiers, array of objects.
	Module 2 (15 hours):
	Polymorphism - Binding-static binding & overloading, constructor overloading function overloading, operator overloading, overloading unary and binary operators. **Modelling Relationships in Class Diagrams:** Association, Aggregation-Composition and examples covering these principles **Inheritance:** Defining base class and its derived class, access specifiers, types of inheritance-single, multiple, hierarchical, multilevel, hybrid inheritance, friend function and friend class, constructors in derived classes. **Modelling Relationships:** Generalization-Specialization and examples covering these principles **Run time Polymorphism -** Dynamic Binding, Function overriding, virtual function, pure virtual function, virtual base class, abstract class. **Pointers:** Introduction to pointers, * and & operators, assigning addresses to pointer variables, accessing values using pointers, pointers to objects & this pointer, pointers to derived classes **File Handling:** File Stream classes, opening and closing file-file opening modes, text file handling, binary file handling. **Applying OOP to solve real life applications:** To cover case studies like library management, order management etc. to design classes covering all relationships
10	**Text Books** 1. Object Oriented Programming with C++, Balagurusamy E., 8th Edition, McGraw Hill Education India. 2. UML & C++: A Practical Guide to Object Oriented Development, Lee/Tepfenhart, Pearson Education, 2nd Edition2015

<table>
<tr><td>11</td><td colspan="2">Reference Books
1. Mastering C++ by Venugopal, Publisher: McGraw-Hill Education, 2017
2. Let Us C++ by KanetkarYashwant, Publisher: BPB Publications, 2020
3. Object Oriented Analysis and Design by Timothy Budd TMH, 2001</td></tr>
<tr><td>12</td><td>Internal Continuous Assessment: 40%</td><td>Semester End Examination: 60%</td></tr>
<tr><td>13</td><td>Continuous Evaluation through:
Class Test on Module 1: 10 marks
Class Test on Module 2: 10 marks
Average of 2 Class Tests: 10 marks
Assignment on Module 1: 5 marks
Assignment on Module 2: 5 marks
Total of 2 Assignments: 10 marks
Total: 20 marks</td><td>Evaluation through:
A Semester End Theory Examination of 1 hour duration for 30 marks as per the paper pattern given below.
Total: 30 marks</td></tr>
<tr><td>14</td><td colspan="2">Format of Question Paper:

Total Marks: 30 Duration: 1 Hour
<table>
<tr><th>Question</th><th>Based On</th><th>Options</th><th>Marks</th></tr>
<tr><td>Q. 1</td><td>Module 1</td><td>Any 2 out of 4</td><td>10</td></tr>
<tr><td>Q. 2</td><td>Module 2</td><td>Any 2 out of 4</td><td>10</td></tr>
<tr><td>Q. 3</td><td>Module 1 & 2</td><td>Any 2 out of 4</td><td>10</td></tr>
</table></td></tr>
</table>