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>