Lesson Plan

Name of the Faculty :Mrs. Reena Rani

Discipline : Computer Engineering

Semester : IV

Subject: OBJECT ORIENTED PROGRAMMING USING C++

Lesson Plan Duration :15 weeks (From January, 2018 to April,2018)

Work Load (Lecture /Practical) per week (in hours): Lectures – 03, Practical – 06)

Week	Theory		Practical	
	Lecture Day	Topic (Including Assignment/ Test)	Practical Day	Topic
1st	1st	**Unit 1: Introduction and Features** 1 Fundamentals of object oriented programming – procedure oriented programming Vs. object oriented programming (OOP)	1st	Write a function using variables as arguments to swap the values of a pair of integers
	2nd	Object oriented programming concepts – Classes, reusability, encapsulation, inheritance, polymorphism, dynamic binding, message passing, Data Hiding	2nd	Write a function using variables as arguments to swap the values of a pair of integers. Revison/Practice Session
	3rd	Benefits of OOPs and its Application	3rd	Consider a shopping list of items for which we place an order with a dealer every month. The list includes such as the code number and price of each item .we would like to perform operations such as adding an item to the list, deleting an item from the list and printing the total value of the order.
			4th	Consider a shopping list of items for which we place an order with a dealer every month. The list includes such as the code number and price of each item .we would like to perform operations such as adding an item to the list, deleting an item from the list and printing the total value of the order. Revison/Practice Session
2nd	1st	**Unit 2: Language Constructs** Review of constructs of C used in C++: variables, types and type declarations, user defined data types	1st	1. Write a program to read name, roll no ,internal external marks using classes and display the same on the screen.
	2nd	increment and decrement operators, relational and logical operators	2nd	1. Write a program to read name, roll no ,internal external marks using classes and display the same on the screen. 2. Revison/Practice Session
	3rd	if then else clause; conditional expressions, input and output statement	3rd	Write a program of swapping of numbers by accessing private numbers using friend function.
			4th	Write a program of swapping of numbers by accessing private numbers using friend function.Revison/Practice Session
3rd	1st	loops, switch case, arrays, structure, unions, functions, pointers; preprocessor directives and Header Files	1st	Define a class to represent a bank account using constructor including the following members:- Data members i) For Single Customer ii) For n Customers a) Name of the depositors b) Account number c) Type of account d) Balance amount in the account Member function - To assign initial values - To deposit an amount - To withdraw an amount after checking the balance

				- To display the name and balance.
	2nd	Scope Resolution Operator Managing Console I/O Operations; C++ Stream	2nd	Define a class to represent a bank account using constructor including the following members:- Data members i) For Single Customer ii) For n Customers a) Name of the depositors b) Account number c) Type of account d) Balance amount in the account Member function - To assign initial values - To deposit an amount - To withdraw an amount after checking the balance - To display the name and balance. Revison/Practice Session
	3rd	Unformatted and Formatted Console I/O	3rd	Create 2 classes OM and DB which store the value of distance. DM store distances in Meters and cm and DB in feet and inches. Write a program that can read values for the class objects and add 1 object OM with another object of DB. Use a friend function to carry out the addition operation the object that stores the results may be a DM object or a DB object, depending upon the units in which the results are required. The display should be in the format of feet and inches or meters and cms depending on the object on display.
			4th	Create 2 classes OM and DB which store the value of distance. DM store distances in Meters and cm and DB in feet and inches. Write a program that can read values for the class objects and add 1 object OM with another object of DB. Use a friend function to carry out the addition operation the object that stores the results may be a DM object or a DB object, depending upon the units in which the results are required. The display should be in the format of feet and inches or meters and cms depending on the object on display. Revison/Practice Session
4th	1st	Revision of Unit 1 and Unit 2	1st	A book shop maintains the inventory of books that are being sold at the shop the list includes details such as author, title and publisher and stock position. Whenever a customer wants the book, the sales person inputs the title and author and the system search the list and display whether it is available ornot. If it is not, a appropriate message is displayed, if it is, then the system displays the book details and requests for the number of copies require. If the requested are available, the total cost of the required copies is displayed: otherwise the message" Required copies not in stock"is displayed. Design a system using a class called books with suitable member functions and constructors. Use new operator in constructor to allocate memory space require. Revision/Practice Session
	2nd	**Unit 3: Classes and Objects** Creation, accessing class members, Private Vs Public	2nd	A book shop maintains the inventory of books that are being sold at the shop the list includes details such as author, title and publisher and stock position. Whenever a customer wants the book, the sales person inputs the title and author and the system search the list and display whether it is available ornot. If it is not, a appropriate message is displayed, if it is, then the system displays the book details and requests for the number of copies require. If the requested are available, the total cost of the required copies is displayed: otherwise the message" Required

				copies not in stock"is displayed. Design a system using a class called books with suitable member functions and constructors. Use new operator in constructor to allocate memory space require.
	3rd	Constructor and Destructor with and without Arguments, Objects	3rd	**Previous program Continued**
			4th	**Previous program Continued**
5th	1st	Dynamic memory Allocation with new and Delete Operator	1st	Define a class string that could work as a user defined string type include constructors that will enable us to create an .un-initialized string String s1; :/ string with length 0 And also to initialize an object with string constant at the time of creation like String s2("well done"); . Include a function that adds two strings to make a third string.
	2nd	**Revision**	2nd	Define a class string that could work as a userdefined string type include constructors that will enable us to create an .un-initialized string String s1; :/ string with length 0 And also to initialize an object with string constant at the time of creation like String s2("well done"); . Include a function that adds two strings to make a third string. Revision/Practice Session
	3rd	**Revision**	3rd	**Previous program Continued**
			4th	**Previous Program Continued**
6th	1st	**Unit 4: Member Function**	1st	Create a class float that contains 2 float data member. Over load all the 4 arithmetic operators so that do operate on the objects of float.
	2nd	Method definition	2nd	Create a class float that contains 2 float data member. Over load all the 4 arithmetic operators so that do operate on the objects of float. Revision/Practice Session
	3rd	Inline Implementation, Constant member functions	3rd	**Previous Program Continued**
			4th	**Previous Program Continued**
7th	1st	Static Function, This Pointer	1st	Programming Exercise on Hybrid Inheritance
	2nd	Friend Function and its Characteristics	2nd	Programming Exercise on Hybrid Inheritance Revision/Practice Session
	3rd	**Revision**	3rd	**Previous Program Continued**
			4th	**Previous Program Continued**
8th	1st	**Unit 5: Overlaoding Member Function** Introduction to Operator Overloading, Need of operator overloading	1st	Define 2 classes POLAR and RECTANGLE to represent points in the POLAR and RECTANGLE systems. Use conversion routines to convert from one system to the other.
	2nd	prefix and postfix, overloading binary operators instream/outstream operator overloading	2nd	Define 2 classes POLAR and RECTANGLE to represent points in the POLAR and RECTANGLE systems. Use conversion routines to convert from one system to the other. Revision/Practice Session
	3rd	Constructor Overloading, Type Conversion, Rules of Operator Overloading	3rd	**Previous Program Continued**

			4th	**Previous Program Continued**
9th	1st	Comparison between Function Overloading and overriding	1st	**Previous Program Continued**
	2nd	**Unit 6: Inheritance** Definition of inheritance, Types of inheritance; Single inheritance, hierarchical inheritance, multiple inheritance, hybrid inheritance	2nd	**Previous Program Continued**
	3rd	protected data, private data, public/data, inheriting constructors and destructors	3rd	Create a base class called shape. use this class to store two double type values that could be used to compute the area of fig. Derive the specific class called TRIANGLE and RECTANGLE from the data shape. Add to base class, a member function get - data ( ) to initialize base class data members and another member and another member function display – area( ) to compute and display the area of the fig.. Make display – area ( ) as a virtual function and redefine function in the derived classes to suit their requirements, Using these 3 classes design a program that will accept dimension of RECTANGLE or TRIANGLE interactivity and display the area. Remember the 2 values given as input will be treated as length of 2 sides in the case of rectangle and as base and height in the case of triangles and used as follows: Area of rectangle = x*y Area of triangle = 1/2 *x*y
			4th	Create a base class called shape. use this class to store two double type values that could be used to compute the area of fig. Derive the specific class called TRIANGLE and RECTANGLE from the data shape. Add to base class, a member function get - data ( ) to initialize base class data members and another member and another member function display – area( ) to compute and display the area of the fig.. Make display – area ( ) as a virtual function and redefine function in the derived classes to suit their requirements, Using these 3 classes design a program that will accept dimension of RECTANGLE or TRIANGLE interactivity and display the area. Remember the 2 values given as input will be treated as length of 2 sides in the case of rectangle and as base and height in the case of triangles and used as follows: Area of rectangle = x*y Area of triangle = 1/2 *x*y Revision/Practice Session
10th	1st	constructor for virtual base classes, constructors and destructors of derived classes, and virtual functions	1st	**Previous Program Continued**
	2nd	size of a derived class, order of invocation,	2nd	**Previous Program Continued**
	3rd	**Revision**	3rd	**Previous Program Continued**
			4th	**Previous Program Continued**
11st	1st	**Unit 7:Polymorphism and Virtual Functions** Importance of virtual function, function call binding	1st	**Previous Program Continued**
	2nd	virtual functions	2nd	**Previous Program Continued**
	3rd	implementing late binding	3rd	**Previous Program Continued**
			4th	**Previous Program Continued**
12th	1st	need for virtual functions, abstract base classes	1st	Exercise on file handling
	2nd	pure virtual functions,	2nd	Exercise on file handling
	3rd	virtual destructors	3rd	Exercise on file handling
			4th	Exercise on file handling

13th	**1st**	**Revision**	**1st**	Exercise on file handling
	2nd	**Unit 8: File and Stream** Components of a file, different operation of the file	**2nd**	Exercise on file handling
	3rd	communication in files, creation of file streams	**3rd**	Exercise on file handling
			4th	Exercise on file handling
14th	**1st**	stream classes, headerfiles, updating of file,	**1st**	Exercise on file handling
	2nd	opening and closing a file	**2nd**	Exercise on file handling
	3rd	file modes and filepointers and their manipulations	**3rd**	Exercise on file handling
			4th	Exercise on file handling
15th	**1st**	functions manipulation of file pointers, detecting end-of-file.	**1st**	**Revision**
	2nd	**Revision**	**2nd**	**Revision**
	3rd	**Revision**	**3rd**	**Revision**
			4th	**Revision**