SCHEME OF EVALUATION

(Scoring Indicators)

Revision : 2015 Course Code : 3134

Course Title : OBJECT ORIENTED PROGRAMMING THROUGH C++

Qn. No		Split up Score	Sub total	Total
	PART- A			
I(1)	List any two – 1. Array, 2.function, 3. Pointer, 4. reference	1+1	2 Marks	10 Marks
I(2)	*type* function-name(*argument-list*); where *type* is valid data type and *argument-list* contains data type and names of arguments passed to function.	2	2 Marks	
I(3)	The wrapping up of data and functions into a single unit(called class) is known as encapsulation	2	2 Marks	
I(4)	List any two – 1. Class member access operator (...*) 2. Scope resolution operator (::) 3. Size operator (size of) 4. Conditional operator (?:)	1+1	2 Marks	
I(5)	To provide means to detect and report an "exceptional circumstance" so that appropriate action can be taken.	2	2 Marks	
	PART- B			
II(1)	Compare any three points- Array • Homogeneous aggregate • An array is collection of items stored at continuous memory locations • An array is a collection of variables of same data type. • **Syntax** type array_name[size]; Structure • Heterogeneous aggregate • Structure elements may not be stored in a contiguous memory location. • A structure is a collection of variables of different data type. • struct sruct_name{ type element1;. } variable1, variable2, . .;	3+3	6 Marks	

II(2)	Explain any three points- 1. Object and class- A Class is a user-defined data-type which has data members and member functions. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. 2. Data abstraction- Abstraction means displaying only essential information and hiding the details. 3. Encapsulation. Encapsulation is defined as wrapping up of data and information under a single unit. 4. Inheritance. The capability of a class to derive properties and characteristics from another class is called Inheritance. 5. Polymorphism.- as the ability of a message to be displayed in more than one form 6. Dynamic Binding: In dynamic binding, the code to be executed in response to function call is decided at runtime. 7. Message Passing: Objects communicate with one another by sending and receiving information to each other.	3 x 2	6 Marks	42 Marks

II(3)	#include <iostream.h>
	using namespace std;
	class person
	{
	private:
	char name[30];
	int age;
	public:
	void getdata(void);
	void display(void);
	};
	void person :: getdata(void)
	{
	cout <<"Enter name and age";
	cin >> name>>age;
	}
	void person :: display(void)
	{
	cout << "\nName"<<name;
	cout <<" \nAge"<<age;
	}
	int main()
	{
	person p;
	p.getdata();
	p.display();
	return 0;
	}
		6	6 Marks	

II(4)	Explain Constructor definition, Syntax, types (Any two)
	Definition: Constructors are special class functions which performs initialization of every object. The Compiler calls the Constructor whenever an object is created. Constructors initialize values to object members after storage is allocated to the object..
	Syntax
	class A
	{ public:
	int x;
	A() //constructor
	{ //object initialisation
	}
	}
	Constructors are of three types:
	• Default Constructor.
	• Parametrized Constructor.
	• Copy Constructor.
		3x2	6 Marks

II(5)	Access Specifiers Public - Class member can be accessible from everywhere, we can access them inside the class, outside the class and in child class also. Private- Class members can be accessible inside the class. Private members are not obtainable to the derived class directly and are inaccessible out of the class scope. Protected - Class members can be accessible only within the class and in the derived classes and used in inheritance.	3x2	6 Marks

II(6)	Explain two points - Definition, example
	
	Composition is the process of making one class using data member of another class
	
	Composition is has-a relation. Composition refers to the relationship between multiple objects. It refers to how objects are related to each other and how they are using each other's functionality.
	
	In composition if an Object owns another object and another object cannot exist without the owner object. Consider the case of Human having a heart. Here Human object contains the heart and heart cannot exist without Human.
	
	Example of Composition
	Class B
	{
	};
	Class A
	{
	B b;
	};
		2 x 3	6 marks

II(7)	The **input operator**, known as the extraction operator (>>), is used with the standard input stream, cin. • cin treats data as a stream of characters. These characters flow from cin to the program through the input operator • The input operator works on two operands,- the cin stream on its left and a variable on its right. • The input operator takes (extracts) the value through cin and stores it in the variable. The **output operator**, called the insertion operator (<<), is used with the standard output stream cout. • cout also treats data as a stream of characters and it flow from the program to cout through the output operator. • The output operator works on two operands - the cout stream on its left and the expression to be displayed on its right. • The output operator directs (inserts) the value to cout. eg. int a; cin>>a; a=a+1; cout<<a;	3x2	6 marks

PART- C				
III(a)	Explain any three points- An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. 1. Arithmetic Operators 2. Relational Operators 3. Logical Operators 4. Bitwise Operators 5. Assignment Operators 6. Other Operators- size of, Condition ? X : Y comma(,),dot(.), arrow(->), &,dot(.) etc.	3x3=9	15 marks	
III(b)	Explain any two points- 1. conditional statements: if, if-else statements 2. loops: while, do-while, for statements 3. jumps: goto, break, continue, return statements 4. multientry code: switch statements with case branches	3x2=6		

IV(a)	#include <iostream.h>
	using namespace std;
	struct student
	{
	char name[50];
	int roll;
	float marks;
	} s[15];
	
	int main()
	{
	int n;
	cout<<"Enter how many students?";
	cin>>n;
	for(i=0;i<n;i++)
	{
	cout<<"Enter name: ";
	cin>> s[i].name;
	
	cout<<"Enter roll number: ";
	cin>> s[i].roll;
	
	cout<<"Enter marks: ";
	cin>> s[i].marks;
	}
	cout<<"Display students Information";
	
	for(i=0;i<n;i++)
	{
		9		120 Marks

	cout<<"\nName: "<< s[i].name; cout<<"\nRoll number: "<< s[i].roll; cout<<"\nMarks: "<< s[i].marks; } return 0; }		

IV(b)	**new operator** request for memory allocation on the Heap. If sufficient memory is available, new operator initializes the memory and returns the address of the newly allocated and initialized memory to the pointer variable.
	Syntax pointer-variable = new data-type;
	Here, pointer-variable is the pointer of type data-type. Data-type could be any built-in data type including array or any user defined data types including structure and class.
	eg. int *p = new int;
	Delete operator is used to destroy array and non-array (pointer) objects which are created by new expression.
	• Delete operator de allocates memory from heap.
	• Pointer to object is not destroyed; value or memory block pointed by pointer is destroyed.
	eg. int* ptr1 = new int;
	delete ptr1; // Destroying ptr1
		3x2=6	15 Marks

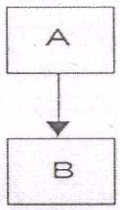
V(a)	• An inline function is a combination of macro & function. At the time of declaration or definition, function name is preceded by word inline • When inline functions are used, the overhead of function call is eliminated. Instead, the executable statements of the function are copied at the place of each function call. And is done by the compiler. #include <iostream> using namespace std; inline int sqr(int x) {	7	

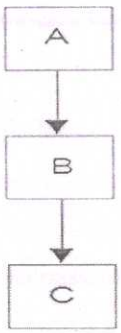
	int y;
	y = x * x;
	return y;
	}
	int main()
	{
	int a =3, b;
	b = sqr(a);
	cout <<b;
	return 0;
	}
	Here, the statement b = sqr(a) is a function call to sqr().The compiler replaces the statement with the executable statement of the function (b = a *a)		

V(b)	Explain any two- Three parameter-passing modes in C++: by value, by pointer, and by reference. 1. Copy of actual parameter is passed to function definition When a parameter is passed by value, the changes made to the parameter within the function do not affect the value of the actual argument used in the function call. 2. Address of actual parameter is passed to function definition When a parameter is passed by pointer or by reference, changes made to the parameter do affect the actual arguments in the client space.	4x2=8	15 marks
VI(a)	• If any classes have multiple functions with same names but different parameters then they are said to be overloaded. • Function overloading allows to use the same name for different functions, to perform, either same or different functions in the same class. • It enhance the readability of the program. #include<iosteam> using namesapce std; int mul (int a, int b)	5+4=9	

	{
	cout << a*b;
	}
	
	int mul(int a, int b, int c)
	{
	cout << a*b*c;
	}
	int main()
	{
	mul (5,4); // mul( ) with 2 parameter will be called
	mul(3,2,9); //mul( ) with 3 parameter will be called
	}		

VI(b)	Explain any three points: Limitations 1. Only existing operators can be overloaded. New operators cannot be created 2. We cannot change the basic meaning of an operator. 3. We cannot change the number of operators. 4. We cannot change operator precedence. 5. We cannot change the a associativity of operator 6. Some operator cannot be overloaded (::, sizeof etc.)	3x2=6	15 Marks

VII(a)	1. Definition,
	2. Syntax and explanation,
	3. example
	Single Inheritance- Definition: A derived class with only one base class is called single inheritance.
	 
	
	Syntax
	class A // Base class
	{
	// BODY OF CLASS A
	};
	class B : acess_specifier A // Derived class of A
	{
	// BODY OF CLASS B
	};
		7	15 Marks

	Multilevel Inheritance-Definition: A derived class with one base class and that base class is a derived class of another is called multilevel inheritance.
	Syntax
	 
	
	class A // Base class
	{
	// BODY OF CLASS A
	};
	class B : acess_specifier A // Derived class of A
	{
	// BODY OF CLASS B
	};
	class C : access_specifier B // Derived from derived class B
	{
	// BODY OF CLASS C
	};
		8		

VIII(a)	Example : minus (-) operator can be overloaded for prefix as well as postfix usage.
	#include <iostream>
	using namespace std;
	
	class Distance {
	private:
	int feet; // 0 to infinite
	int inches; // 0 to 12
	
	public:
	// required constructors
	Distance() {
	feet = 0;
	inches = 0;
	}
	Distance(int f, int i) {
	feet = f;
	inches = i;
	}
		9		

```
    // method to display distance
    void displayDistance() {
      cout << "F: " << feet << " I:" << inches <<endl;
    }

    // overloaded minus (-) operator
    Distance operator- () {
      feet = -feet;
      inches = -inches;
      return Distance(feet, inches);
    }
};

int main() {

Distance D1(11, 10), D2(-5, 11);

  -D1;                  // apply negation
  D1.displayDistance();  // display D1

  -D2;                  // apply negation
  D2.displayDistance();  // display D2

  return 0;
}
```

VIII(b)	Explain 2 points, Definition, example Definition: Private members are accessed only within the class they are declared. Friend function is used to access the private and protected members of different classes. It works as bridge between classes. • Friend function must be declared with **friend** keyword. • Friend function must be declare in all the classes from which we need to access private or protected members. • Friend function will be defined outside the class without specifying the class name. • Friend function will be invoked like normal function, without any object	3+3=6	15 Marks

	Example:
	class Box {
	double width;
	public:
	double length;
	friend void printWidth(Box box); // friend function
	void setWidth(double wid);
	};			

IX(a)	Explain 3 points - definition,explanation of syntax,example • Exception handling is the process of handling errors and exceptions in such a way that they obstruct or delay normal execution of the system. • For example, User divides a number by zero, this will compile successfully but an exception or run time error will occur due to which our applications will be crashed. • Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling uses three keywords: try, catch, throw. • **throw** − A program throws an exception when a problem shows up. This is done using a **throw** keyword. • **catch** − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception. • **try** − A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks. example	2+4+4 = 10	15 Marks	
IX(b)	Explain two points – definition and explanation , example A Cast operator is an unary operator which forces one data type to be converted into another data type. C++ supports four types of casting: 1. Static Cast 2. Dynamic Cast 3. Const Cast 4. Reinterpret Cast	5		

Example:

```
#include <iostream>
using namespace std;
int main()
{
   float f = 3.5;

   // using cast operator
   int b = static_cast<int>(f);

   cout << b;
}
```

X(a)

```
#include<iostream>
usingnamespacestd;

// template function
template<class T>
T Large(T n1, T n2)
{
        return(n1 > n2)? n1 : n2;
}

int main()
{
        int i1, i2;
```

9

```
    float f1, f2;
    char c1, c2;

    cout << "Enter two integers:\n";
    cin >> i1 >> i2;
    cout << Large(i1, i2) <<" is larger." << endl;

    cout << "\nEnter two floating-point numbers:\n";
    cin >> f1 >> f2;
    cout << Large(f1, f2) <<" is larger." << endl;

    cout << "\nEnter two characters:\n";
    cin >> c1 >> c2;
    cout << Large(c1, c2) << " has larger ASCII value.";

    return 0;
}
```

X(b)

List any 3 points -

1. Virtual functions must be members function of some class.
2. Virtual functions cannot be static members.
3. They are accessed through object pointers. (object of class)
4. They can be a friend of another class.
5. A virtual function must be defined in the base class. even though it is not used.
6. The prototypes of a virtual function of the base class and all the derived classes must be identical.
7. Virtual functions are declared with the keyword virtual.
8. Virtual function takes a different functionality in the derived class

3x2=6

15 Marks