WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Object Oriented Programming Using C++ **Subject Code:** **22316**

Important Instructions to examiners:

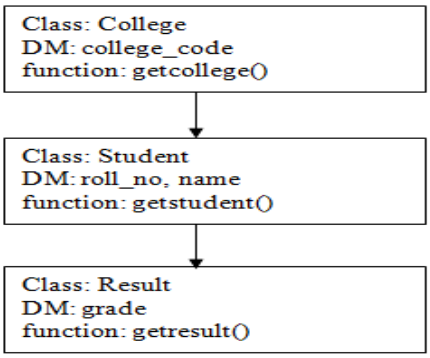
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling errors should not be given more Importance (Not applicable for subject English and Communication Skills).
4) While assessing figures, examiner may give credit for principal components indicated in the figure. The figures drawn by candidate and model answer may vary. The examiner may give credit for any equivalent figure drawn.
5) Credits may be given step wise for numerical problems. In some cases, the assumed constant values may vary and there may be some difference in the candidate's answers and model answer.
6) In case of some questions credit may be given by judgement on part of examiner of relevant answer based on candidate's understanding.
7) For programming language papers, credit may be given to any other program based on equivalent concept.

<table>
<tr><th>Q. No.</th><th>Sub Q.N.</th><th>Answer</th><th>Marking Scheme</th></tr>
<tr><td>1.</td><td>a)
Ans.</td><td>Attempt any <u>FIVE</u> of the following:
State the difference between OOP and POP.
<table>
<tr><th>Sr. No.</th><th>OBJECT ORIENTED PROGRAMMING (OOP)</th><th>PROCEDURE ORIENTED PROGRAMMING (POP)</th></tr>
<tr><td>1</td><td>Focus is on data rather than procedure.</td><td>Focus is on doing things (procedure).</td></tr>
<tr><td>2</td><td>Programs are divided into multiple objects.</td><td>Large programs are divided into multiple functions.</td></tr>
<tr><td>3</td><td>Data is hidden and cannot be accessed by external functions.</td><td>Data move openly around the system from function to function.</td></tr>
<tr><td>4</td><td>Objects communicate with each other through function.</td><td>Functions transform data from one form to another by calling each other.</td></tr>
<tr><td>5</td><td>Employs bottom-up approach in program design</td><td>Employs top-down approach in program design.</td></tr>
</table></td><td>10
2M

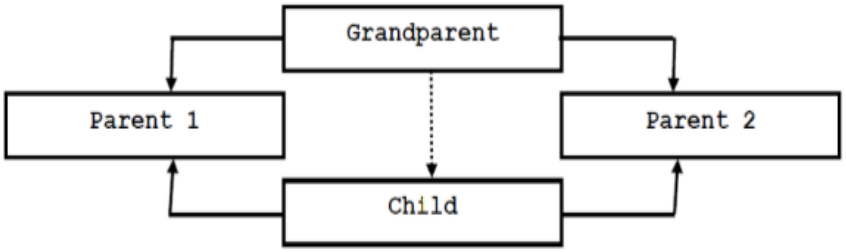
Any two differences 1M each</td></tr>
</table>

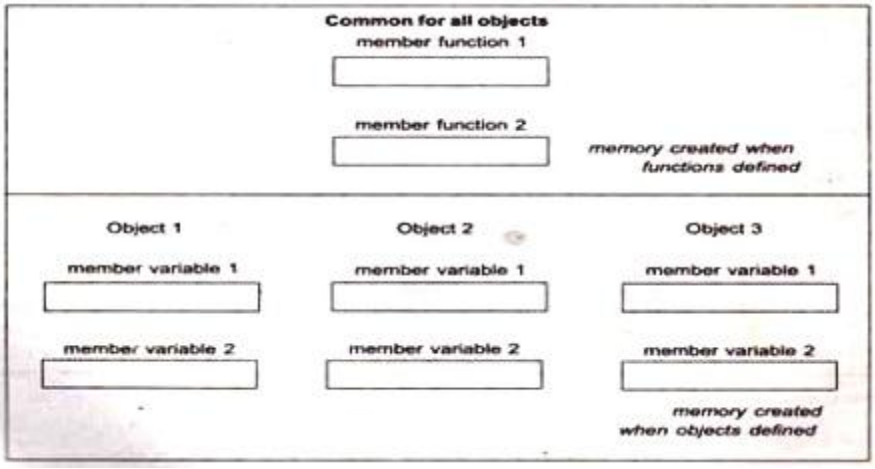
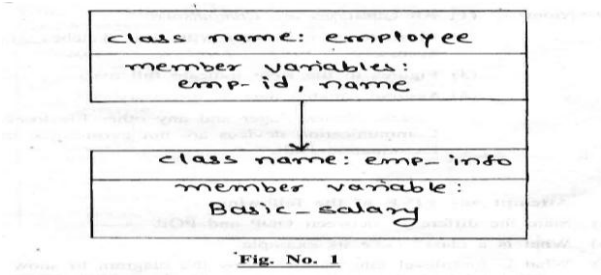
		6 | Object oriented approach is used in C++ language. | Procedure oriented approach is used in C language.	

	b) **Ans.**	**What is a class? Give its example.**
		Class is a user defined data type that combines data and functions together. It is a collection of objects of similar type.
		
		Example:
		class Student
		{
		int rollno;
		char name[10];
		public:
		void getdata( );
		void putdata( );
		};
		
			2M
			Class definition 1M
			
			Correct example 1M

	c) **Ans.**	**What is multilevel inheritance? Draw the diagram to show multilevel inheritance. using classes with data member and member function.** When a class is derived from another derived class then it is called as multilevel inheritance.  	**2M** ***Define multilevel inheritance 1M*** ***Diagram 1M***
	d) **Ans.**	**Explain use of scope resolution operator.** It is used to uncover a hidden variable. Scope resolution operator allows access to the global version of a variable. The scope resolution operator is used to refer variable of class anywhere in program. **::** Variable_name **OR** Scope resolution operator is also used in classes to identify the class	**2M** ***Correct use 2M***

		to which a member function belongs. Scope resolution operator is used to define function outside of class.
		Return_type class_name**::** function_name( )
		{
		Function body
		}	

	e) Ans.	**Write two properties of static member function.** i) A static member function can have access to only other static data members and functions declared in the same class. ii) A static member function can be called using the class name with a scope resolution operator instead of object name as follows: class_name::function_name;	**2M** ***Two properties 1M each***
	f) Ans.	**Explain virtual base class with suitable example.** A virtual base class (Grandparent class) is a class that avoids duplication of inherited data in derived class (child class) derived from parent classes (parent1 and parent2) which in turn derived from base class. ***Example:***  **Fig. a: Virtual Base Class**	**2M** ***Explanation of Virtual base class 1M*** ***Example 1M***
	g) Ans.	**Give syntax and use of fclose ( ) function.** **Syntax:** int fclose(FILE* stream); **Use:** This function is used to close a file stream. The data that is buffered but not written is flushed to the OS and all unread buffered data is discarded.	**2M** ***Syntax 1M*** ***Correct use 1M***
2.	**a) Ans.**	**Attempt any <u>THREE</u> of the following:** **Describe memory allocation for objects.** The memory space for object is allocated when they are declared and not when the class is specified. The member functions are created and placed in memory space only once when they are defined as a part of	**12** **4M**

		a class definition. Since all the objects belonging to that class use the same member functions, no separate space is allocated for member functions. When the objects are created only space for member variable is allocated separately for each object. Separate memory locations for the objects are essential because the member variables will hold different data values for different objects.  **Fig: Memory allocation for objects**	***Description 2M*** ***Diagram 2M***
	b)	**Write a program to implement single inheritance from the following Refer Figure No.1**  Fig. No. 1	**4M**

	Ans.	***(Note: Any other correct logic shall be considered)***
		#include<iostream.h>
		#include<conio.h>
		class employee
		{
		protected:
		int emp_id;
		char name[10];
		};
			Class declaration 1M each

		class emp_info:public employee
		{
		int basic_salary;
		public:
		void getdata()
		{
		cout<<"Enter emp id";
		cin>>emp_id;
		cout<<"Enter name";
		cin>>name;
		cout<<"Enter basic salary";
		cin>>basic_salary;
		}
		void putdata()
		{
		cout<<"\nEmp_id="<<emp_id;
		cout<<"\nName="<<name;
		cout<<"\nBasic Salary="<<basic_salary;
		}
		};
		void main()
		{
		emp_info e;
		clrscr();
		e.getdata();
		e.putdata();
		getch();
		}
			Function declaration 1M
			
			Main function 1M

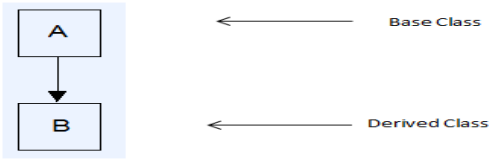
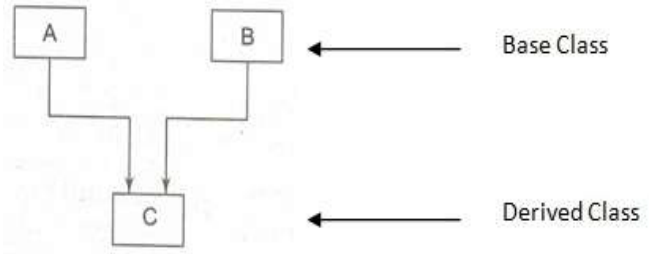
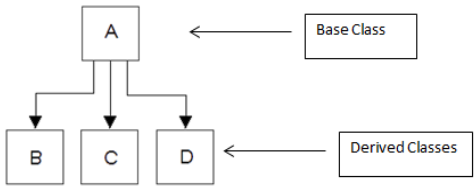
	c) Ans.	**Write any four benefits of OOP.** **Benefits of OOP:** 1. We can eliminate redundant code and extend the use of existing classes. 2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity. 3. The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program. 4. It is possible to have multiple instances of an object to co-exist without any interference. 5. It is possible to map objects in the problem domain to those in the	**4M** ***Any four benefits 1M each***

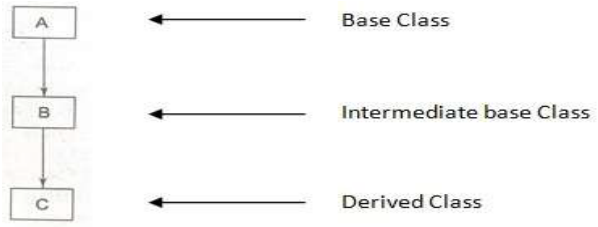
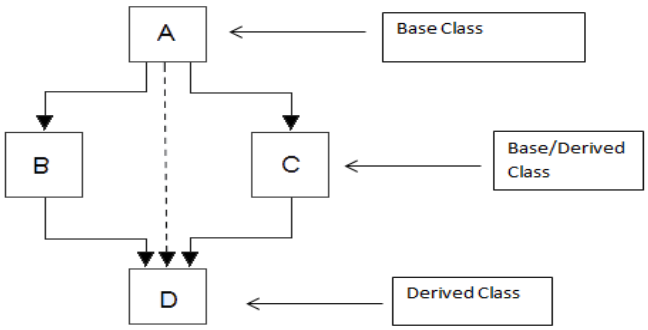
		program. 6. It is easy to partition the work in a project based on objects. 7. The data-centered design approach enables us to capture more details of a model in implementable form. 8. Object-oriented systems can be easily upgraded from small to large systems. 9. Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler. 10. Software complexity can be easily managed.	

	d) Ans.	**Describe 'this' pointer with an example.**
		'this' pointer:
		C++ uses a unique keyword called 'this' to represent an object that invokes a member function. This unique pointer is automatically passed to a member function when it is invoked. 'this' is a pointer that always point to the object for which the member function was called.
		For example, the function call A.max ( ) will set the pointer 'this' to the address of the object A. Then suppose we call B.max ( ), the pointer 'this' will store address of object B.
		
		Example:
		#include<iostream.h>
		class sample
		{
		int a;
		public:
		void setdata(int x)
		{
		this ->a=x;
		}
		void putdata()
		{
		cout<<this ->a;
		}
		};
		void main()
		{
		sample s;
			4M
			
			Description 2M
			
			Correct example 2M

		s.setdata(100);
		s.putdata( );
		}
		In the above example, this pointer is used to represent object s when setdata ( ) and putdata ( ) functions are called.	

3.		**Attempt any <u>THREE</u> of the following:**	**12**
	a) **Ans.**	**Write the applications of object oriented programming.** **Applications of object oriented programming are:** 1) Real time systems 2) Simulation and modeling 3) Object-oriented databases 4) Hypertext, hypermedia and expertext 5) AI and expert systems 6) Neural networks and parallel programming 7) Decision support and office automation systems 8) CIM/CAM/CAD systems	**4M** ***Any four correct applications 1M each***
	b) **Ans.**	**State the rules for writing destructor function.** **Rules for writing destructor function are:** 1) A destructor is a special member function which should destroy the objects that have been created by constructor. 2) Name of destructor and name of the class should be same. 3) Destructor name should be preceded with tilde (~) symbol. 4) Destructor should not accept any parameters. 5) Destructor should not return any value. 6) Destructor should not be classified in any types. 7) A class can have at most one destructor.	**4M** ***Any four correct rules 1M each***
	c) **Ans.**	**What is inheritance? Give different types of inheritance.** **Inheritance:** The mechanism of deriving new class from an old/existing class is called inheritance. **OR** Inheritance is the process by which objects of one class acquired the properties of objects of another classes. ***Syntax:*** class derived-class-name: visibility-mode base-class-name { ------//	**4M** ***Correct explanation of inheritance 2M***

		-----// members of derived class -----// }; **Types of inheritance:** **1) Single inheritance**: In single inheritance, a derived class is derived from only one base class. **Diagram:**  **2) Multiple inheritance**: In multiple inheritance, derived class is derived from more than one base classes. **Diagram:**  **3) Hierarchical inheritance**: In hierarchical inheritance, more than one derived classes are derived from single class. **Diagram:**  **4) Multilevel inheritance**: In multilevel inheritance, a derived class is derived from a derived class (intermediate base class) which in turn	***Correct types of inheritance (any 4)*** **2M**

		derived from a single base class. **Diagram:**  **5) Hybrid inheritance**: Hybrid inheritance is a combination of single, multiple, multilevel and hierarchical inheritance. **Diagram**:  	
	d) Ans.	**What are the rules for virtual function?** **Rules for virtual function:** 1. The virtual functions must be members of some class. 2. They cannot be static members. 3. They are accessed by using object pointers. 4. A virtual function can be a friend of another class. 5. A virtual function in a base class must be defined, even though it may not be used. 6. The prototypes of the base class version of a virtual function and all the derived class versions must be identical. 7. We cannot have virtual constructors, but we can have virtual destructors. 8. While a base pointer can point to any type of the derived object, the reverse is not true. 9. When a base pointer points to a derived class, incrementing or	**4M** ***Any four rules 1M each***

		decrementing it will not make it to point to the next object of the derived class. 10. If a virtual function is defined in the base class, it need not be necessarily redefined in the derived class.	

4.	**a)**
	Ans.
		Attempt any <u>THREE</u> of the following:
		What is parameterized constructor?
		A constructor that accepts parameters is called as parameterized constructor.
		In some applications, it may be necessary to initialize the various data members of different objects with different values when they are created. Parameterized constructor is used to achieve this by passing arguments to the constructor function when the objects are created.
		
		Example:
		class ABC
		{
		int m;
		public:
		ABC(int x)
		{
		m=x;
		}
		void put()
		{
		cout<<m;
		}
		};
		void main()
		{
		ABC obj(10);
		obj.put();
		}
		In the above example, constructor ABC (int x) is a parameterized constructor function that accepts one parameter. When 'obj' object is created for class ABC, parameterized constructor will invoke and data member m will be initialized with the value 10 which is passed as an argument. Member function put ( ) displays the value of data member 'm'.
			12
			4M
			
			Correct descripti on 4M

	b)
	
	Ans.
		Write a program to sort an 1-d array in ascending order.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int arr[20];
		int i, j, temp,n;
		clrscr();
		cout<<"\n Enter the array size:";
		cin>>n;
		cout<<"\n Enter array elements:";
		for(i=0;i<n;i++)
		{
		cin>>arr[i];
		}
		for(i=0;i<n;i++)
		{
		for(j=i+1;j<n;j++)
		{
		if(arr[i]>arr[j])
		{
		temp=arr[i];
		arr[i]=arr[j];
		arr[j]=temp;
		}
		}
		}
		cout<<"Sorted Array:";
		for(i=0;i<n;i++)
		{
		cout<<"\n"<<arr[i];
		}
		getch();
		}
			4M
			
			Correct array input 1M
			
			Sorting of 1D array in ascending order 2M
			
			Display of sorted array 1M

	c) **Ans.**	**Explain the friend function with proper example.** **Friend function:** The private members of a class cannot be accessed from outside the class but in some situations two classes may need access of each	**4M**

		other's private data. So a common function can be declared which can be made friend of more than one class to access the private data of more than one class. The common function is made friendly with all those classes whose private data need to be shared in that function. This common function is called as friend function. Friend function is not in the scope of the class in which it is declared. It is called without any object. The class members are accessed with the object name and dot membership operator inside the friend function. It accepts objects as arguments.
		
		Example:
		Program to interchange values of two integer numbers using friend function.
		#include<iostream.h>
		#include<conio.h>
		class B;
		class A
		{
		int x;
		public:
		void accept()
		{
		cout<<"\n Enter the value for x:";
		cin>>x;
		}
		friend void swap(A,B);
		};
		class B
		{
		int y;
		public:
		void accept()
		{
		cout<<"\n Enter the value for y:";
		cin>>y;
		}
		friend void swap(A,B);
		};
		void swap(A a,B b)
			Correct explanation of friend function 2M
			
			Correct example 2M

		{
		cout<<"\n Before swapping:";
		cout<<"\n Value for x="<<a.x;
		cout<<"\n Value for y="<<b.y;
		int temp;
		temp=a.x;
		a.x=b.y;
		b.y=temp;
		cout<<"\n After swapping:";
		cout<<"\n Value for x="<<a.x;
		cout<<"\n Value for y="<<b.y;
		}
		void main()
		{
		A a;
		B b;
		clrscr();
		a.accept();
		b.accept();
		swap(a,b);
		getch();
		}	

	d)
	
	Ans.
		Write a program to count the number of lines in file.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<fstream.h>
		#include<conio.h>
		void main()
		{
		ifstream file;
		char ch;
		int n=0;
		clrscr();
		file.open("abc.txt");
		while(file)
		{
		file.get(ch);
		if(ch=='\n')
		n++;
			4M
			
			Opening of file 1M
			
			Counting number of lines 2M
			
			Printing number of lines in a file 1M

		}
		cout<<"\n Number of lines in a file are:"<<n;
		file.close();
		getch();
		}	

5.		**Attempt any TWO of the following:**	**12**
	a)	**Write a program to declare a class 'student' having data members as 'stud_name' and 'roll_no'. Accept and display this data for 5 students.** ***(Note: Any other correct logic shall be considered)***	**6M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class student
		{
		int roll_no;
		char stud_name[20];
		public:
		void Accept();
		void Display();
		};
		void student::Accept()
		{
		cout<<"\n Enter student's name and roll no\n";
		cin>>stud_name>>roll_no;
		}
		void student::Display()
		{
		cout<<stud_name<<"\t"<<roll_no<<"\n";
		}
		void main()
		{
		student S[5];
		inti;
		clrscr();
		for(i=0;i<5;i++)
		{
		S[i].Accept();
		}
		cout<<"Student details \n Student's Name \t Roll No\n";
			Class declaration 2M
			
			Accept ( )1M
			
			Display ( ) 1M
			
			Main ( ) with array 2M

```
for(i=0;i<5;i++)
{
 S[i].Display();
}
getch();
}
```

b) Ans. **State and explain the visibility modes used in inheritance.** **6M**

Visibility modes:

- private
- protected
- public

Base class visibility	Derived class visibility		
	Private	Protected	Public
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Private	Protected	Protected
Public	Private	Protected	Public

- **Private:**
  - When a base class is privately inherited by a derived class, 'public members' and 'protected members' of the base class become 'private members' of the derived class.
  - Therefore, the public and protected members of the base class can only be accessed by the member functions of derived class but, cannot be accessed by the objects of the derived class.

Syntax:

```
class derived: private base
{
//Members of derived class;
 };
```

Explanation 2M for each visibility mode

- **Public:**
  - When a base class is publicly inherited by a derived class then 'protected members' of base class becomes 'protected members' and 'public members' of the base class become 'public members' of the derived class.
  - Therefore the public members of the base class can be accessed by both the member functions of derived class as well

		as the objects of the derived class.
		Syntax:
		class derived: public base
		{
		//Members of derived class;
		};
		
		• **Protected:**
		o When a base class is protectedly inherited by a derived class, 'public and protected members' of the base class become 'protected members' of the derived class.
		o Therefore the public and protected members of the base class can be accessed by the member functions of derived class as well as the member functions of immediate derived class of it but they cannot be accessed by the objects of derived class
		Syntax:
		class derived: protected base
		{
		//Members of derived class;
		};	

	c)
	
	Ans.
		Write a program to declare a class 'book' containing data members as 'title', 'author-name', 'publication', 'price'. Accept and display the information for one object using pointer to that object.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		class book
		{
		char author_name[20];
		char title[20];
		char publication[20];
		float price;
		public:
		void Accept();
		void Display();
		};
		void book::Accept()
		{
			6M
			
			Class declaration 2M
			
			Accept () 1M

		cout<<"\n Enter book's title, author_name, publication and price \n:";
		cin>> title >>author_name>> publication >> price;
		}
		void student::Display()
		{
		cout<<title <<"\t"<<author_name<<"\t"<<publication <<"\t"<<
		price<<"\n"<<;
		}
		void main()
		{
		book b, *p;
		clrscr();
		p=&b;
		p->Accept();
		cout<<"title \t author_name \t publication \t price\n";
		p-> Display();
		getch();
		}
			Display ( ) 1M
			
			Main( ) with pointer 2M

6.	**a)**
	
	Ans.
		Attempt any <u>TWO</u> of the following:
		Write a program that copies contents of one file into another file.
		(Note: Any other correct logic shall be considered)
		Assuming input file to be copied file1.txt contents are "Hello Friends…" and file where the contents need to copy is file2.txt already created
		
		#include<iostream.h>
		#include<conio.h>
		#include<fstream.h>
		#include<stdio.h>
		#include<stdlib.h>
		void main()
		{
		clrscr();
		ifstream fs;
		ofstream ft;
		char ch, fname1[20], fname2[20];
		cout<<"Enter source file name with extension (like files.txt) : ";
		gets(fname1);
		fs.open(fname1);
			12
			6M
			
			File open and close 2M

```
if(!fs)
{
        cout<<"Error in opening source file..!!";
        getch();
        exit(1);
}
cout<<"Enter target file name with extension (like filet.txt) : ";
gets(fname2);
ft.open(fname2);
if(!ft)
{
        cout<<"Error in opening target file..!!";
        fs.close();
        getch();
        exit(2);
}
while(fs.eof()==0)
{
        fs>>ch;
        ft<<ch;
}
cout<<"File copied successfully..!!";
fs.close();
ft.close();
getch();
}
```

Logic for copy contents 4M

b) **Write a program to implement the following hierarchy using suitable member functions. Refer Figure No.2.** **6M**

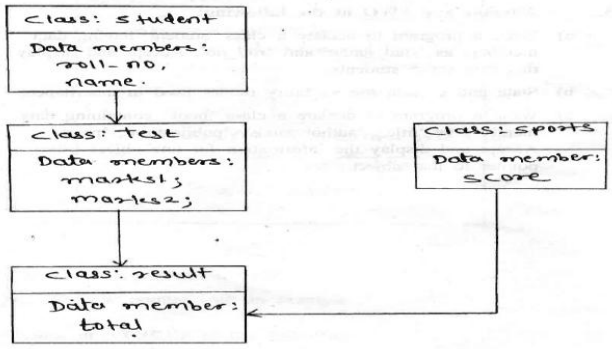


Fig. No. 2

	Ans.	***(Note: Any other correct logic shall be considered)***
		# include <iostream.h>
		#include<conio.h>
		class Student
		{
		int roll_no;
		char name[10];
		public:
		void read_studentData()
		{
		cout<<"Enter student's roll no and name \n";
		cin>>roll_no>> name;
		}
		void display_studentData ()
		{
		cout<<"\n roll_no\t name\n";
		cout<<roll_no<<"\t"<<name<<"\n";
		}
		};
		class test: public Student
		{
		protected:
		int marks1,marks2;
		public:
		void read_test()
		{
		cout<<"\n Enter test marks\n";
		cin>>marks1>>marks2;
		}
		
		void display_test()
		{
		cout<<"\n test Marks \n Marks1 \t Marks2 \n";
		cout<<marks1<<"\t"<<marks2;
		}
		};
		class sports
		{
		int score;
			Class student declaration 1M
			
			Class test declaration 1M

```
public:
void read_sportsData()
{
cout<<"\n Enter sport score\n";
cin>> score;
}
void display_sportsData()
{
        cout<<"\n sport score:"<<score;
}
};
class result: public test, public sports
{
int total;
public:
        void read_result()
        {
        read_ studentData () ;
        read_test();
        read_sportsData();
        total=marks1+marks2;
}
        void display_result()
        {
        display_ studentData () ;
        display_test();
        display_sportsData();
        cout<<"\n Total="<<total;
}
};
void main()
{
        result r;
        clrscr();
        r.read_result();
        r.display_result();
        getch();
}
```

Class sports declaration 1M

Class result declaration 2M

Main () 1M

WINTER – 2019 EXAMINATION
MODEL ANSWER

Subject: Object Oriented Programming Using C++ **Subject Code: 22316**

	c)
	
	Ans.
		Write a program to overload the '—' unary operator to negate the values.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		#include<string.h>
		class Number
		{
		int x,y;
		public:
		Number (int a, int b)
		{
		a =x;
		b =y;
		}
		void display()
		{
		cout<<"value of x="<<x<<"\n Value of y= "<<y;
		}
		void operator - ( )
		{
		x = - x;
		y = - y;
		}
		};
		void main ()
		{
		Number N1(5,6);
		clrscr ();
		N1. display ();
		-N1;
		cout<<"\n After negation:";
		N1. display ();
		getch ();
		}
			6M
			
			Correct Program with output 6M