V2V EDTECH LLP

Online Coaching at an Affordable Price.

OUR SERVICES:

- Diploma in All Branches, All Subjects
- Degree in All Branches, All Subjects
- BSCIT / CS
- Professional Courses

☎ +91 93260 50669

▶ V2V EdTech LLP

v2vedtech.com

v2vedtech

SUMMER – 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.

Q. No.	Sub Q.N.	Answer	Marking Scheme
1.	 **a)** **Ans.**	**Attempt any FIVEof the following:** **State the use of cin and cout.** **cin**: cin is used to accept input data from user (Keyboard). **cout:**cout is used to display output data on screen.	**10** **2M** ***Use - 1M each***

	b)
	Ans.
		Describe derived class with example.
		Derived class: In inheritance a new class is derived from an old class. The new class is referred as derived class. The derived class can inherit all or some properties of its base class.
		Example:
		class base
		{
		};
		class derived: public base
		{
		};
			2M
			Description 1M
			Example 1M

	c) **Ans.**	**State 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	**2M** ***Use 2M***

		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 to which a member function belongs. Scope resolution variable is used to define function outside of class.
		Return_typeclass_name**::** function_name( )
		{
		}	

	d) Ans.	**Define class and object.** **Class:** Class is a user defined data type that combines data and functions together. It is a collection of objects of similar type. **Object:** It is a basic run time entity that represents a person, place or any item that the program has to handle.	**2M** ***Definition 1M each***
	e) Ans.	**Write the use of ios : : in and ios : : out.** **ios::in** - It is used as file opening mode to specify open file reading only. **ios::out**- It is used as file opening mode to specify open file writing only.	**2M** ***Each use 1M***
	f) Ans.	**Describe use of static data member.** **Use of static data member:** Static data member (variable) is used to maintain values common to the entire class. Only one copy of static member is created for the entire class and is shared by all the objects of that class. Its lifetime is the entire program.	**2M** ***Use 2M***

	g)
	
	Ans.
		Give meaning of following statements:
		int *ptr, a = 5;
		ptr = & a;
		cout<< * ptr;
		cout<< (* ptr) + 1;
		int *ptr, a = 5;
		Declare pointer variable ptr and variable a with initial value 5.
		ptr = & a;
		initialize pointer variable with address of variable a (store address of variable a in ptr)
		cout<< * ptr;
			2M
			Meaning of each Statement $^{1/2}$M

		Displays value of a i.e. value at address stored inside ptr. It displays value 5. **cout<< (* ptr) + 1;** Displays value by adding 1 to the value at address stored inside ptr. It displays value 6	

2.	**a)**
	
	Ans.
		Attempt any <u>THREE</u> of the following:
		Write a 'C++' program to find factorial of given number using loop.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int no,fact=1,i;
		clrscr();
		cout<<"Enter number:";
		cin>>no;
		for(i=1;i<=no;i++)
		{
		fact=fact*i;
		}
		cout<<"Factorial ="<<fact;
		getch();
		}
			12
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	b)
	
	Ans.
		Write a C++ program to declare a class COLLEGE with members as college code. Derive a new class as STUDENT with members as studid. Accept and display details of student along with college for one object of student.
		(Note: Any other correct logic shall be considered)
		
		#include<iostream.h>
		#include<conio.h>
		class COLLEGE
		{
		protected:
		int collegecode;
		};
			4M
			
			Definition of class COLLEGE: 1M

		class STUDENT:public COLLEGE
		{
		int studid;
		public:
		void accept()
		{
		cout<<"Enter college code:";
		cin>>collegecode;
		cout<<"Enter student id";
		cin>>studid;
		}
		void display()
		{
		cout<<"College code:"<<collegecode;
		cout<<"Student id:"<<studid;
		}
		};
		void main()
		{
		STUDENT s;
		clrscr();
		s.accept();
		s.display();
		getch();
		}
			Definition of class STUDENT 1M
			Accept and display function 1M
			Main function 1M

	c)
	Ans.
		Write a C++ program to find smallest number from two numbers using friend function. (Hint: use two classes).
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		class class2;
		class class1
		{
		int no1;
		public:
		void get1()
		{
		cout<<"Enter number 1:";
		cin>>no1;
			4M
			Definition of class1 1M

```
}
friend void smallest(class1 no1,class2 no2);
};
class class2
{
int no2;
public:
void get2()
{
cout<<"Enter number 2:";
cin>>no2;
}
friend void smallest(class1 no1,class2 no2);
};
void smallest(class1 c1,class2 c2)
{
if(c1.no1<c2.no2)
cout<<"no1 is smallest";
else
cout<<"no2 is smallest";
}
void main()
{
class1 c1;
class2 c2;
clrscr();
c1.get1();
c2.get2();
smallest(c1,c2);
getch();
}
```

Definition of class2 1M

Friend function 1M

Main() function 1M

d) **Ans.** **Differentiate between run time and compile time polymorphism.** **4M**

Any four differences 1M each

Sr. No.	Compile time polymorphism	Runtime polymorphism
1	In this polymorphism, an object is bound to its function call at compile time.	In this polymorphism, selection of appropriate function is done at run time.

2	Functions to be called are known well before.	Function to be called is unknown until appropriate selection is made.
3	This does not require use of pointers to objects	This requires use of pointers to object
4	Function calls execution are faster	Function calls execution are slower
5	It is implemented with operator overloading or function overloading	It is implemented with virtual function.

3. **Attempt any THREE of the following:** **12**

a) **Write a C++ program to create a class STUDENT** **4M**
The data members of STUDENT class.
Roll_No
Name
Marks
(Note: Accepting and displaying data functions is optional).

Ans.

```
#include<iostream.h>
#include<conio.h>
class STUDENT
{
 int Roll_No;
 char Name[20];
 float Marks;
};
```

Correct Class definition with data member declaration: 4M

OR

```
#include<iostream.h>
#include<conio.h>
class STUDENT
{
 int Roll_No;
 char Name[20];
 float Marks;
 public:
 void Accept();
 void Display();
};
void STUDENT::Accept()
```

```
{
cout<<"\nEnter data of student:";
cout<<"\nRoll number:";
cin>>Roll_No;
cout<<"\nName:";
cin>>Name;
cout<<"\nMarks:";
cin>>Marks;
}
void STUDENT::Display()
{
cout<<"\nStudents data is:";
cout<<"\nRoll number:"<<Roll_No;
cout<<"\nName:"<<Name;
cout<<"\nMarks:"<<Marks;
}
void main()
{
 STUDENT S[5];
int i;
clrscr();
 for(i=0;i<5;i++)
 {
  S[i].Accept();
 }
 for(i=0;i<5;i++)
 {
  S[i].Display();
 }
getch();
}
```

b) **Accept data for five students and display it. Write a C++ program to displya sum of array elements of array size n.** **4M**

(Note: Any other correct logic shall be considered)

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{
int arr[20],i,n,sum=0;
```

		clrscr();
		cout<<"\nEnter size of an array:";
		cin>>n;
		cout<<"\nEnter the elements of an array:";
		for(i=0;i<n;i++)
		 {
		cin>>arr[i];
		 }
		 for(i=0;i<n;i++)
		 {
		  sum=sum+arr[i];
		 }
		cout<<"\nArray elements are:";
		 for(i=0;i<n;i++)
		 {
		cout<<arr[i]<<" ";
		 }
		cout<<"\nSum of array elements is:"<<sum;
		getch();
		}
			Initialization of array 2M
			Calculation and display of sum of array elements 2M

	c)
	Ans.
		Describe with examples, passing parameters to base class constructor and derived class constructor by creating object of derived class.
		When a class is declared, a constructor can be declared inside the class to initialize data members. When a base class contains a constructor with one or more arguments then it is mandatory for the derived class to have a constructor and pass arguments to the base class constructor. When both the derived and base classes contain constructors, the base constructor is executed first and then the constructor in the derived class is executed. The constructor of derived class receives the entire list of values as its arguments and passes them on to the base constructors in the order in which they are declared in the derived class.
		
		General form to declare derived class constructor:
		Derived-constructor (arglist1, arglist (D)):Base1(arglist1)
		{
		Body of derived class constructor
		}
			4M
			Correct Description 2M

Derived constructor declaration contains two parts separated with colon (:). First part provides declaration of arguments that are passed to the derived constructor and second part lists the function calls to the base constructors.

Example:

```
#include<iostream.h>
#include<conio.h>
class base
{
int x;
public:
base(int a)
{
x=a;
cout<<"Constructor in base x="<<x;
}
};
class derived: public base
{
int y;
public:
derived(int a,int b):base(a)
{
y=b;
cout<<"Constructor in derived.y="<<y;
}
};
void main()
{
clrscr();
derived ob(2,3);
getch();
}
```

Correct example 2M

In the above example, base class constructor requires one argument and derived class constructor requires one argument. Derived class constructor accepts two values and passes one value to base class constructor.

d) **Describe how memory is allocated to objects of class with suitable diagram.** **4M**

Ans. **Description:**

The memory space for object is allocated when they are declared and not when the class is specified. Actually, the member functions are created and placed in memory space only once when they are defined as a part of 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 this is shown in fig:

Correct description 2M

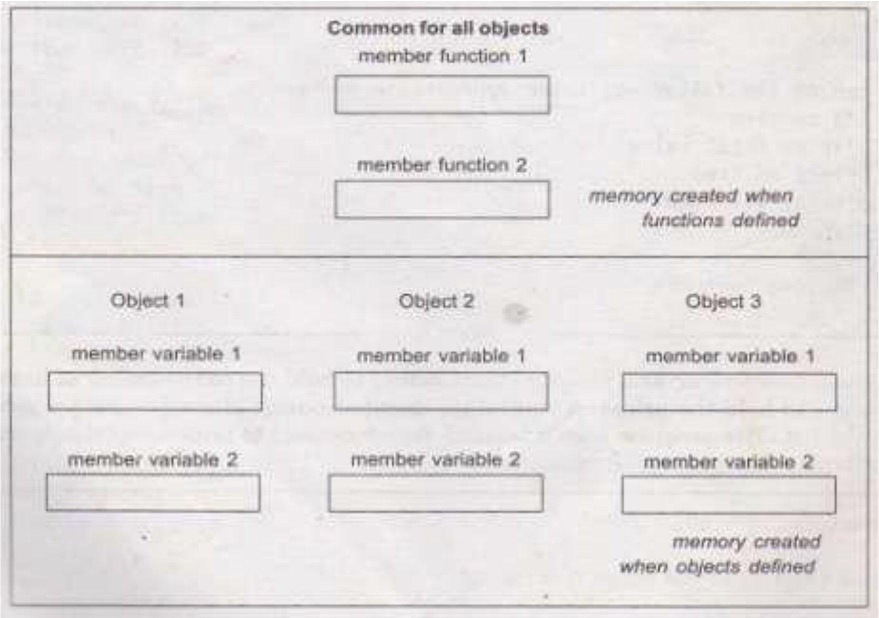


Fig: Memory allocation for objects

Correct diagram for memory allocation of objects 2M

4. **Attempt any <u>THREE</u> of the following:** **12**

a) **Write a program to implement multiple inheritance as shown in following Figure No.1:** **4M**

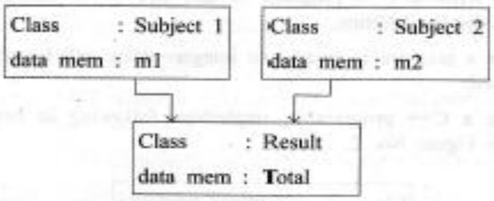


Fig. No. 1

Accept and display data for one object of class result.
(Note: Any other relevant logic should be considered).

Ans. **Program:**

```
#include<iostream.h>
#include<conio.h>
class Subject1
{
 protected:
 float m1;
};
class Subject2
{
 protected:
 float m2;
};
class Result:public Subject1,public Subject2
{
 float Total;
 public:
 void accept()
 {
cout<<"Enter marks of subject1:";
cin>>m1;
cout<<"\nEnter marks of subject2:";
cin>>m2;
 }
 void calculate()
 {
  Total=(m1+m2);
 }
 void display()
 {
```

Definition of Class Subject1 1M

Definition of Class Subjec2 1M

Definition of Class Result 1M

		cout<<"\nSubject 1 marks:"<<m1;
		cout<<"\nSubject 2 marks:"<<m2;
		cout<<"\nTotal is:"<<Total;
		 }
		};
		void main()
		{
		 Result r;
		 clrscr();
		 r.accept();
		 r.calculate();
		 r.display();
		 getch();
		}
			main function 1M

	b) **Ans.**	**Describe following terms: Inheritance, data abstraction, data encapsulation, dynamic binding.** **Inheritance:** 1. Inheritance is the process by which objects of one class acquire the properties of objects of another class. 2. It supports the concept of hierarchical classification. It also provides the idea of reusability. **Data abstraction:** 1. Data abstraction refers to the act of representing essential features without including the background details or explanations. 2. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes. **Data encapsulation:** 1. The wrapping up of data and functions together into a single unit (called class) is known as encapsulation. 2. By this attribute the data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. **Dynamic Binding:** 1. Dynamic binding refers to the linking of a procedure call to be executed in response to the call. 2. It is also known as late binding. It means that the code associated with a given procedure call is not known until the time of the call at run-time.	**4M** ***Correct description 1M each***

c) **State and describe visibility modes and its effects used in inheritance.** 4M

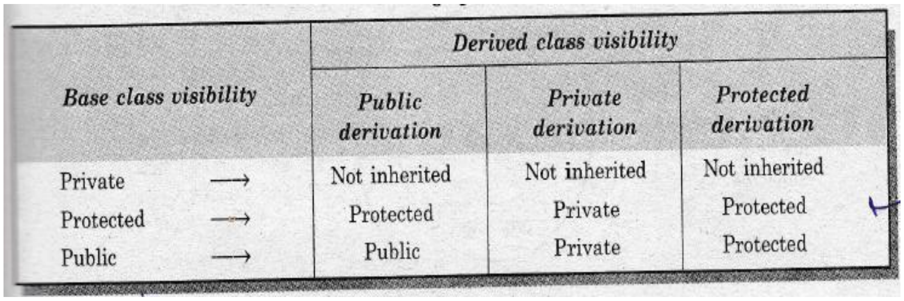
(Note: Diagram is optional)

Ans. Different visibility modes are:

1. Private
2. Protected
3. Public

State visibility modes 1M

Effects of visibility modes in inheritance:



Base class visibility	*Derived class visibility*		
	Public derivation	*Private derivation*	*Protected derivation*
Private ⟶	Not inherited	Not inherited	Not inherited
Protected ⟶	Protected	Private	Protected
Public ⟶	Public	Private	Protected

Private members of base class are not inherited directly in any visibility mode.

1. **Private visibility mode**
   In this mode, protected and public members of base class become private members of derived class.
2. **Protected visibility mode**
   In this mode, protected and public members of base class become protected members of derived class.
3. **Public visibility mode**
   In this mode, protected members of base class become protected members of derived class and public members of base class become public members of derived class.

Description of effect of visibility mode in inheritance 1M each

d) **Write a C++ program to count number of spaces in text file.** 4M

(Note: Any other correct logic shall be considered)

Ans. **Program:**

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
void main()
{
```

```
ifstream file;
int s=0;
 char ch;
clrscr();
file.open("abc.txt");
 while(file)
 {
file.get(ch);
  if(ch==' ')
  {
   s++;
  }
 }
cout<<"\nNumber of spaces in text file are:"<<s;
getch();
}
```

Correct logic 2M

Correct syntax 2M

e) **Differentiate between contractor and destructor.** 4M

(*Note: Contractor shall be considered as Constructor.)*

Ans.

Constructor	Destructor
A constructor is a special member function whose task is to initialize the objects of its class.	A destructor is a special member function whose task is to destroy the objects that have been created by constructor.
It constructs the values of data members of the class.	It does not construct the values for the data members of the class.
It is invoked automatically when the objects are created.	It is invoked implicitly by the compiler upon exit of a program/block/function.
Constructors are classified in various types such as : Default constructor Parameterized constructor Copy constructor Overloaded constructor	Destructors are not classified in any types.
A class can have more than one constructor.	A class can have at the most one constructor.
Constructor accepts	Destructor never accepts any

Any four correct differences 1M each

parameters. Also it can have default value for its parameter.	parameter.
Syntax: classname() {… … }	**Syntax:** destructor name is preceded with tilde. ~classname() {…. …. }

Example:
ABC()
{
…
}
	Example:
	~ABC()
	{
	….
	}

5.		**Attempt any <u>TWO</u> of the following:**	12
	a)	**(i) Write any three rules of operator overloading.** **(ii) Write a program in C++ to overload unary ‘_’ operator to negate values of data members of class.**	6M
	Ans.	**(i) Write any three rules of operator overloading.** **Rules for overloading operators:** 1. Only existing operators can be overloaded. New operators cannot be created. 2. The overloaded operator must have at least one operand that is of user defined data type. 3. We can’t change the basic meaning of an operator. That is to say, we can’t redefine the plus(+) operator to subtract one value from other. 4. Overloaded operators follow the syntax rules of the original operators. They can’t be overridden. 5. There are some operators that can’t be overloaded. 6. We can’t use friend functions to overload certain operators. However, member function scan be used to overload them. 7. Unary operators overloaded by means of member function take no explicit arguments and return no explicit values, but, those overloaded by means of the friend function, take one reference argument (the object of the relevant class). 8. Binary operators overloaded through a member function, take one	***Any three rules of operator overloading 1M each***

explicit argument and those which are overloaded through a friend function take two explicit arguments.

9. When using binary operators overloaded through a member function, the left hand operand must be an object of the relevant class.
10. Binary arithmetic operators such as +,-,* and / must explicitly returna value. They must not attempt to change their own arguments.

(ii) Write a program in C++ to overload unary ‘_’ operator to negate values of data members of class.

(Note: Any other correct logic shall be considered)

Ans.

```
#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<<”\nValue of y= ”<<y;
}
void operator - ( )
{
x = - x;
y = - y;
}
};
void main()
{
Number N1(5,6);
clrscr();
N1.display();
```

Class declaration with member 1M

Operator function definition 1M

Main() function definition 1M

```
-N1;
cout<<"\n After negation:";
N1. display ();
getch();
}
```

b) **Write a C++ program to append data from abc.txt to xyz.txt file.** **6M**

(Note: Any other correct logic shall be considered)

Ans. Assuming input file as abc.txt with contents "World" and output file named as xyz.txt with contents "Hello" have been already created.

```
#include <iostream.h>
#include<fstream.h>

int main()
{
            fstream f;
            ifstream fin;
            fin.open("abc.txt",ios::in);
            ofstream fout;
            fout.open("xyz.txt",  ios::app);
            if (!fin)
           {
               cout<< "file not found";
           }
           else
           {
              fout<<fin.rdbuf();
           }
         char ch;
         f.seekg(0);
         while (f)
        {
          f.get(ch);
          cout<< ch;
        }
        f.close();
        return 0;
}
```

Correct logic 3M

Correct Syntax 3M

Output:

```
Hello World
```

c) **Write a C++ program to declare a class student with members as roll no, name and department. Declare a parameterized constructor with default value for department as ‘CO’ to initialize members of object. Initialize and display data for two students.** **6M**

(Note: Any other relevant logic should be considered).

Ans.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class student
{
int roll_no;
char name[20],department[40];
public:
student(int rno,char *n,char *d="CO")
{
roll_no=rno;
strcpy(name,n);
strcpy(department,d);
}
void display()
{
cout<<"\n Roll No:"<<roll_no;
cout<<"\n Name:"<<name;
cout<<"\n Department:"<<department;
 }
};
void main()
{
student s1(112," Chitrakshi"),s2(114,"Anjali");
clrscr();
s1.display();
s2.display();
getch();
}
```

Class student 1M

Constructor definition with default value 2M

Display function definition 1M

Main function definitio 2M

6. **a)** **Attempt any TWO of the following:** **12**

(i) Describe structure of C++ program with diagram. **6M**

(ii) Write a C++ program to add two 3 x 3 matrices and display addition.

Ans.

(i) Describe structure of C++ program with diagram.

INCLUDE HEADER FILES
DECLARE CLASS
DEFINE MEMBER FUNCTIONS
DEFINE MAIN FUNCTION

Correct diagram 1M

Description:-

1. Include header files

In this section a programmer include all header files which are require to execute given program. The most important file is iostream.h header file. This file defines most of the C++statements like cout and cin. Without this file one cannot load C++ program.

2. Declare Class

In this section a programmer declares all classes which are necessary for given program. The programmer uses general syntax of creating class.

Description 2M

3. Define Member Functions

This section allows programmer to design member functions of a class. The programmer can have inside declaration of a function or outside declaration of a function.

4. Define Main Functions

This section the programmer creates object and call various functions writer within various class.

(ii) Write a C++ program to add two 3 x 3 matrices and display addition.

(Note: Any other relevant logic should be considered).

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{
clrscr();
int mat1[3][3], mat2[3][3], i, j, mat3[3][3];
cout<<"Enter matrix 1 elements :";
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
 cin>>mat1[i][j];
}
}
cout<<"Enter matrix 2 elements :";
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
cin>>mat2[i][j];
}
}
cout<<"Adding the two matrix to form the third matrix\n";
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
mat3[i][j]=mat1[i][j]+mat2[i][j];
}
}
cout<<"The two matrix added successfully...!!";
cout<<"The new matrix will be :\n";
for(i=0; i<3; i++)
{
for(j=0; j<3; j++)
{
cout<<mat3[i][j]<<" ";
}
cout<<"\n";
}
getch();
}
```

Accepting two matrices 1M

Adding two matrices 1M

Displaying addition 1M

b) **Write a program to swap two integers using call by reference method.** **6M**

(Note: Any other relevant logic should be considered).

Ans.

```
#include<iostream.h>
#include<conio.h>
void swap(int*p, int*q)
{
int t;
 t=*p;
 *p=*q;
 *q=t;
}
void main()
{
int a,b;
float x,y;
clrscr();
cout<<"Enter values of a and b\n";
cin>>a>>b;
cout<<"Before swapping\n";
cout<<"a="<<a<<"\tb="<<b<<endl;
 swap(&a, &b);
cout<<"After swapping\n";
cout<<"a="<<a<<"\tb="<<b<<endl;
getch();
}
```

Correct logic 3M

Correct Syntax 3M

c) **Write a C++ program to implement following in heritance. Refer Figure No.2:** **6M**

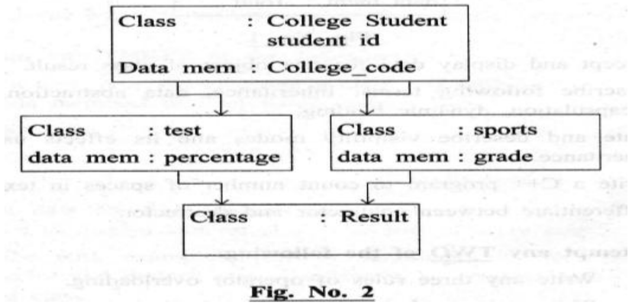


Fig. No. 2

Accept and display data for one object of class result (Hint: use virtual base class).

(Note: Any other relevant logic should be considered).

Ans.

```
# include <iostream.h>
#include<conio.h>
class College_Student
{
int student_id;
char College_code[5];
public:
 void read_collegeStud_Data()
{
cout<<”Enter college code and student id\n”;
cin>>college_code>>student_id;
}
void display_collegeStud_Data()
{
cout<<”\ncollege code\tstudent id\n”;
cout<<college_code<<”\t”<<student_id<<”\n”;
}
};

class test: virtual public College_Student
{
float percentage;
public:
void read_test()
{
cout<<”\n Enter test percentage\n”;
cin>> percentage;
}
void display_test()
{
cout<<”\n test percentage:”<<percentage;
}
};
class sports: virtual public College_Student
{
char grade[5];
public:
void read_sportsData()
```

Each class (four classes) definition 1M

Use of virtual base class 1M

Main function definition 1M

```
{
cout<<”\n Enter sport grade\n”;
cin>> grade;
}
void display_sportsData()
{
Cout<<”\n sport grade:”<<grade;
}
};
class result: public test, public sports
{
public:
void read_result()
{
read_collegeStud_Data() ;
read_test()
read_sportsData();
}
void display_result()
{
display_collegeStud_Data() ;
display_test()
display_sportsData();
}
};
void main()
{
result r;
clrscr();
r.read_result();
r.display_result();
}
```