Boost your Diploma Exams Preparation

Msbte Diploma Exam Papers

GET IT ON Google Play

WINTER – 2018 EXAMINATION
MODEL ANSWER

Subject: Object Oriented Programming with 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 FIVE of the following:** **State any four object oriented languages.** **Object oriented programming language:** • C++ • Smalltalk • Object pascal • java • Simula • Ada • Turbo pascal • Eiffel • C# • Python	**10** **2M** ***Any 4 languages ½ M each***
	b) Ans.	**Describe use of protected access specifier used in the class.** Protected access specifier is use to declare a class member that is accessible by the member functions within its class and any class immediately derived from it.	**2M** ***Correct use 2M***

WINTER – 2018 EXAMINATION
MODEL ANSWER

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

c) Ans — **Differentiate between OOP and POP** — **2M**

Sr. No.	PROCEDURE ORIENTED PROGRAMMING (POP)	OBJECT ORIENTED PROGRAMMING (OOP)
1	Focus is on doing things (procedure).	Focus is on data rather than procedure.
2	Large programs are divided into multiple functions.	Programs are divided into multiple objects.
3	Data move openly around the system from function to function.	Data is hidden and cannot be accessed by external functions.
4	Functions transform data from one form to another by calling each other.	Objects communicate with each other through function.
5	Employs top-down approach in program design.	Employs bottom-up approach in program design
6	Procedure oriented approach is used in C language.	Object oriented approach is used in C++ language.

Any two relevant differences 1M each

d) Ans. — **Write any two characteristics of destructor.** — **2M**

Characteristics:
1. It is used to destroy objects created by a constructor.
2. Name of destructor and name of the class is same.
3. Its name is preceded with tilde (~) symbol.
4. It never takes any argument.
5. It does not return any value.
6. It is invoked implicitly by the compiler upon exit from the program (or block or function) i.e when scope of object is over.

Any two characteristics- 1M each

e) Ans. — **Describe meaning of the following** — **2M**

(i) ios : : in
(ii) ios : : out

(i) ios : : in : It is a file mode. It is used to open a file in read only mode.

Meaning of 'in' 1M

(ii) ios : : out : It is a file mode. It is used to open a file in write only mode.

Meaning of 'out' 1M

WINTER – 2018 EXAMINATION
MODEL ANSWER

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

	f)	**Give output for following code:** **class student** **{** **int roll no;** **char name [14];** **} s[6];** **void main()** **{** **cout<<sizeof(s);** **}**	**2M**
	Ans	Considering roll_no(Single variable) the output is: 96 OR Considering roll, no (Two variables) the output is: 108 OR Considering roll no the output is: error – space between roll and no	***Correct output 2M***

	g)
	Ans
		Write syntax to define a derived class
		Syntax:
		class derived_class_name : visibility_mode/access_specifier base_class_name
		{
		class body
		};
			2M
			Correct syntax 2M

2		**Attempt any <u>THREE</u> of the following**	**12**
	a)	**Write a C++ program to accept array of five elements, find and display smallest number from an array.**	**4M**

	Ans	#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int a[5],smallest,i;
		clrscr();
		cout<<" Enter array elements:";
		for(i=0;i<5;i++)
		cin>>a[i];
		smallest=a[0];
		for(i=1;i<5;i++)
		{
		if(a[i]<smallest)
			Correct logic 2M
			Correct syntax 2M

		{
		smallest=a[i];
		}
		}
		cout<<endl<<"Smallest number="<<smallest;
		getch();
		}	

	b)
	
	Ans
		Write a C++ program to declare a class 'College' with data members as name and college code. Derive a new class 'student' from the class college with data members as sname and roll no. Accept and display details of one student with college data.
		#include<iostream.h>
		#include<conio.h>
		class college
		{
		char name[10];
		int collegecode;
		public:
		void getcollege()
		{
		cout<<"Enter college name:";
		cin>>name;
		cout<<"Enter college code:";
		cin>>collegecode;
		}
		void putcollege()
		{
		cout<<endl<<"College name="<<name;
		cout<<endl<<"College code="<<collegecode;
		}
		};
		class student:public college
		{
		char sname[10];
		int rollno;
		public:
		void getstudent()
		{
		cout<<"Enter student name";
			4M
			
			Declaration and Definition of Base Class 1M
			
			Declaration and Definition of Derived Class 2M
			
			Main function 1M

		cin>>sname;
		cout<<"Enter roll no:";
		cin>>rollno;
		}
		void putstudent()
		{
		cout<<endl<<"Student name:="<<sname;
		cout<<endl<<"Roll no:="<<rollno;
		}
		};
		void main()
		{
		student s;
		clrscr();
		s.getcollege();
		s.getstudent();
		s.putcollege();
		s.putstudent();
		getch();
		}	

	c)
	
	Ans
		Write a C++ program to declare a class 'circle' with data members as radius and area. Declare a function getdata to accept radius and putdata to calculate and display area of circle.
		#include<iostream.h>
		#include<conio.h>
		class circle
		{
		float radius,area;
		public:
		void getdata()
		{
		cout<<"Enter radius:";
		cin>>radius;
		}
		void putdata()
		{
		area=3.14*radius*radius;
		cout<<"Area of circle="<<area;
			4M
			
			Decalaration and Definition of class with functions 3M

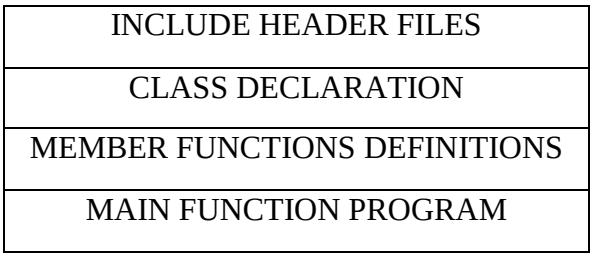
		}
		};
		void main()
		{
		circle c;
		clrscr();
		c.getdata();
		c.putdata();
		getch();
		}
			Main function 1M

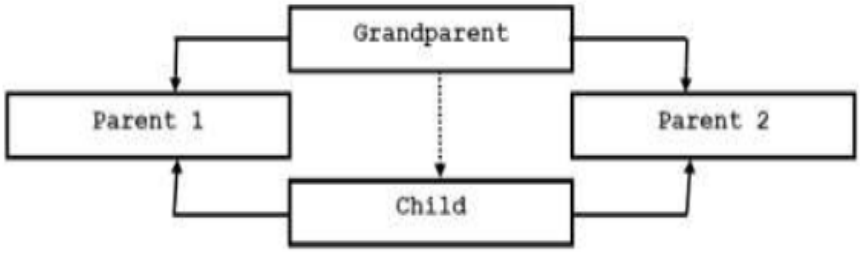
	d)
	
	Ans.
		With suitable example, describe effect of ++ and - - operators used with pointer in pointer arithmetic.
		++ Operator: - It is referred as increment operator that increments the value of variable. If ++ operator is used with pointer variable, then pointer variable points to next memory address that means pointer increment with respect to size of the data type used to declare pointer variable.
		
		Example:-
		int a[5]={10,20,30,40,50},*ptr;
		ptr=a[0];
		for(i=0;i<5;i++)
		{
		cout<<*ptr;
		ptr++;
		}
		In the above example, ptr points to memory location of a[0]. Increment statement ptr++ increments ptr by memory size of int i.e 2 bytes and ptr points to a[1].
		
		- - Operator: - It is referred as decrement operator that decrements the value of variable. If - - operator is used with pointer variable, then pointer variable points to previous memory address that means pointer decrement with respect to size of the data type used to declare pointer variable.
			4M
			
			Description of ++ operator 1M
			
			Any relevant Example 1M
			
			Description of - - operator 1M

		Example:-
		int a[5]={10,20,30,40,50},*ptr;
		ptr=a[4];
		for(i=0;i<5;i++)
		{
		cout<<*ptr;
		ptr- -;
		}
		
		In the above example, ptr points to memory location of a[4]. Decrement statement ptr- - decrements ptr by memory size of int i.e 2 bytes and ptr points to a[3].
			Example 1M

3	**a)**
	
	Ans.
		Attempt any <u>THREE</u> of the following
		Write a C++ program to declare a class addition with data members as x and y. Initialize values of x and y with constructor. Calculate addition and display it using function 'display'.
		#include<iostream.h>
		#include<conio.h>
		class addition
		{
		int x,y;
		public:
		addition(int,int);
		void display();
		};
		addition::addition (int x1,int y1)
		{
		x=x1;
		y=y1;
		}
		void addition::display()
		{
		cout<<"\nAddition of two numbers is:"<<(x+y);
		}
		void main()
		{
		addition a(3,4);
			12
			4M
			
			Declaration and definition of class with constructor and display function 3M
			
			Main function 1M

		a.display();
		getch();
		}	

	b) **Ans**	**With suitable diagram describe structure of C++ program.** General C++ program has following structure.  **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. Class Declaration** In this section a programmer declares all classes which are necessary for given program. The programmer uses general syntax of creating class. **3. Member Functions Definition** 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. Main Function Program** In this section programmer creates objects and calls various functions writer within various class.	**4M** ***Correct diagram 2M*** ***Description 2M***
	c) **Ans.**	**Describe the concept of virtual base class with suitable example.** ***Note: Program/diagram with syntax shall be considered as an example.*** **Virtual Base Class:** An ancestor class is declared as virtual base class which is used to avoid duplication of inherited members inside child class due to multiple path of inheritance.	**4M** ***Description 2M***



Consider a hybrid inheritance as shown in the above diagram. The child class has two direct base classes, Parent1 and Parent2 which themselves have a common base class as Grandparent. The child inherits the members of Grandparent via two separate paths. All the public and protected members of Grandparent are inherited into Child twice, first via Parent1 and again via Parent2. This leads to duplicate sets of the inherited members of Grandparent inside Child class. The duplication of inherited members can be avoided by making the common base class as virtual base class while declaring the direct or intermediate base classes as shown below.

```
class Grandparent
{
};
class Parent1:virtual public Grandparent
{
};
class Parent2:virtual public Grandparent
{
};
class Child: public Parent1,public Parent2
{
};
```

Example 2M

Example

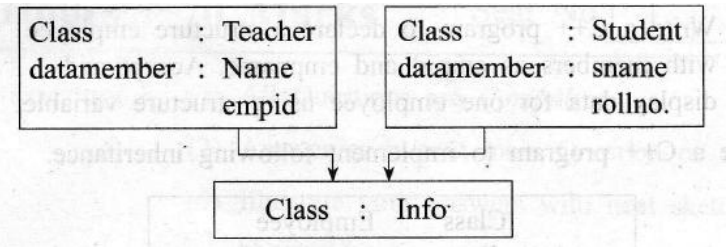
```
#include<iostream.h>
#include<conio.h>
class student
{
int rno;
```

```
public:
void getnumber()
{
cout<<"Enter Roll No:";
cin>>rno;
}
void putnumber()
{
cout<<"\n\n\t Roll No:"<<rno<<"\n";
}
};
class test: virtual public student
{
public:
int part1,part2;
void getmarks()
{
cout<<"Enter Marks\n";
cout<<"Part1:";
cin>>part1; cout<<"Part2:";
cin>>part2;
}
void putmarks()
{
cout<<"\t Marks Obtained\n";
cout<<"\n\t Part1:"<<part1;
cout<<"\n\tPart2:"<<part2;
}
};
class sports: public virtual student
{
public:
int score;
void getscore()
{
cout<<"Enter Sports Score:";
cin>>score;
}
void putscore()
```

		{
		cout<<"\n\t Sports Score is:"<<score;
		}
		};
		class result: public test, public sports
		{
		int total;
		public:
		void display()
		{
		total=part1+part2+score;
		putnumber();
		putmarks();
		putscore();
		cout<<"\n\t Total Score:"<<total;
		}
		};
		void main()
		{
		result obj;
		clrscr();
		obj.getnumber();
		obj.getmarks();
		obj.getscore();
		obj.display();
		getch();
		}	

	d) Ans	**Describe use of static data member in C++ with example.** **Use of static data member**: Static data member is used to maintain values common to the entire class. It is initialized to zero when the first object of its class is created. Only one copy of that member is created for the entire class and is shared by all the objects of that class. **Example:** #include<iostream.h> #include<conio.h> class test {	**4M** ***Use of static data member 2M*** ***Relevant example 2M***

```
static int count;
int obj_no;
public:
void getdata()
{
obj_no=++count;
cout<<"\n Object number="<<obj_no;
}
static void showcount()
{
cout<<"\n total number of objects="<<count;
}
};
int test::count;
void main()
{
test t1,t2;
clrscr();
t1.getdata();
t2.getdata();
test::showcount();
test t3;
t3.getdata();
test::showcount();
getch();
}
```

4	**a)**	**Attempt any <u>THREE</u> of the following** **Write a C++ program to implement inheritance shown in following figure:**  **Accept and display data of one teacher and one student using object of class 'Info'** ***Note: Any other correct logic of multiple inheritance in program shall be considered.***	**12** **4M**

	Ans	#include<iostream.h>
		#include<conio.h>
		class Teacher
		{
		protected:
		char Name[20];
		int empid;
		};
		class Student
		{
		protected:
		char sname[20];
		int rollno;
		};
		class Info:public Teacher,public Student
		{
		public:
		void acceptT()
		{
		cout<<"\nEnter data for teacher:";
		cout<<"\nName:";
		cin>>Name;
		cout<<"\nEmployee id:";
		cin>>empid;
		}
		
		void displayT()
		{
		cout<<"\nTeacher's data is:";
		cout<<"\nName:"<<Name;
		cout<<"\nEmployee id:"<<empid;
		}
		void acceptS()
		{
		cout<<"\nEnter student's data:";
		cout<<"\nName:";
		cin>>sname;
			Correct definition of class - Teacher 1M
			
			Correct definition of class- Student 1M
			
			Correct definition of class- Info 1M

		cout<<"\nRoll no:";
		cin>>rollno;
		}
		void displayS()
		{
		cout<<"\nStudent's data is:";
		cout<<"\nName:"<<sname;
		cout<<"\nRoll no:"<<rollno;
		}
		};
		void main()
		{
		Info I;
		clrscr();
		I.acceptT();
		I.displayT();
		I.acceptS();
		I.displayS();
		getch();
		}
			Correct definition of main function 1M

	b)
	
	Ans
		Write a C++ program to print multiplication table of 7. (example: 7 x 1 ….7 x 10 = 70)
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int num;
		clrscr();
		cout<<"Multiplication table for 7 is:"<<endl;
		for(num=1;num<=10;num++)
		{
		cout<<"7 *"<<num<<"="<<7*num<<endl;
		}
		getch();
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

	c)	**Write a C++ program to swap two integer numbers and swap two float numbers using function overloading.**	**4M**

		(Hint: overload swap function) ***Note: Any other relevant logic shall be considered.***	

	Ans	#include<iostream.h>
		#include<conio.h>
		void swap(int a,int b)
		{
		int temp;
		temp=a;
		a=b;
		b=temp;
		cout<<"\nInteger values after swapping are:"<<a<<" "<<b;
		}
		void swap(float x,float y)
		{
		float temp1=x;
		x=y;
		y=temp1;
		cout<<"\nFloat values after swapping are:"<<x<<" "<<y;
		}
		void main()
		{
		clrscr();
		swap(10,20);
		swap(10.15f,20.25f);
		getch();
		}
			Correct logic 2M
			
			Correct syntax 2M

	d)	**Write a C++ program to count number of spaces present in contents of file.** ***Note: Any other relevant logic shall be considered***	**4M**
	Ans	#include<iostream.h> #include<fstream.h> #include<conio.h> void main() { ifstream file; charch;	***Correct logic 2M*** ***Correct***

		int s=0;
		clrscr();
		file.open("abc.txt");
		while(file)
		{
		file.get(ch);
		if(ch==' ')
		{
		s++;
		}
		}
		cout<<"\nNumber of spaces present in the content of the given file are:"<<s;
		getch();
		}
			syntax 2M

	e)
	
	Ans.
		Write a C++ program to find greatest number among two numbers from two different classes using friend function.
		#include<iostream.h>
		#include<conio.h>
		class second;
		class first
		{
		int x;
		public:
		void getx()
		{
		cout<<"\nEnter the value of x:";
		cin>>x;
		}
		friend void max(first,second);
		};
		class second
		{
		int y;
		public:
		void gety()
		{
		cout<<"\nEnter the value of y:";
			4M
			
			Correct definition of class first 1M
			
			Correct definition of class second

		cin>>y;
		}
		friend void max(first,second);
		};
		void max(first a,second b)
		{
		if(a.x>b.y)
		{
		cout<<"\Greater value is:"<<a.x;
		}
		else
		{
		cout<<"\nGreater value is:"<<b.y;
		}
		}
		void main()
		{
		first a;
		second b;
		clrscr();
		a.getx();
		b.gety();
		max(a,b);
		getch();
		}
			1M
			
			Correct definition of friend function 1M ,
			
			Correct definition of main function 1M

5	**a)** **Ans**	**Attempt any <u>TWO</u> of the following** **Write a C++ program to overload binary operator '+' to concatenate two strings.** #include<iostream.h> #include<conio.h> #include<string.h> class opov { char str1[10]; public: void getdata() {	**12** **6M** ***Creating Class 2M*** ***Operator Functio***

		cout<<"\nEnter a strings";
		cin>>str1;
		}
		void operator +(opov o)
		{
		cout<<strcat(str1,o.str1);
		}
		};
		void main()
		{
		opov o1,o2;
		clrscr();
		o1.getdata();
		o2.getdata();
		o1+o2;
		getch();
		}
			n
			2M
			
			Main Functio n
			2M

	b)
	
	Ans
		Write a C++ program to write 'Welcome to poly' in a file. Then read the data from file and display it on screen.
		Note: Any other relevant logic shall be considered
		#include<iostream.h>
		#include<conio.h>
		#include<fstream.h>
		void main()
		{
		char str[25] = "Welcome to poly",ch;
		clrscr();
		ofstream fout;
		fout.open("output.txt");
		fout<<str;
		fout.close();
		ifstream fin;
		fin.open("output.txt");
		while (!fin.eof())
		{
		fin.getline(str, 25);
		cout<<str<<endl;
		}
			6M
			
			Writing data in file 3M
			
			Reading data from file and display on screen 3M

		fin.close();
		getch();
		}	

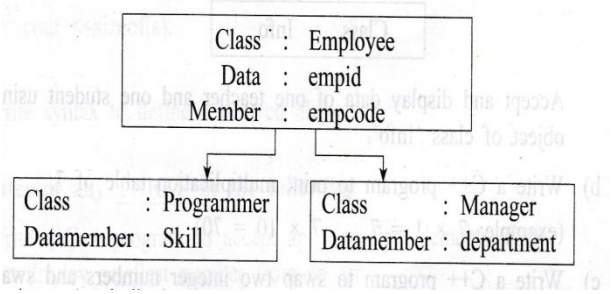
	c)
	
	Ans
		Write a C++ program to declare a class 'Account' with data members as accno, name and bal. Accept data for eight accounts and display details of accounts having balance less than 10,000.
		#include<iostream.h>
		#include<conio.h>
		class Account
		{
		long int accno, bal;
		char name[10];
		public:
		void getdata()
		{
		cout<<"\nEnter account number, balance and name ";
		cin>>accno>>bal>>name;
		}
		void putdata()
		{
		if(bal>10000)
		{
		cout<<"\nThe Account Number is "<<accno;
		cout<<"\nThe Balance is "<<bal;
		cout<<"\nThe Name is "<<name;
		}
		}
		};
		void main()
		{
		Account a[8];
		int i;
		clrscr();
		for(i=0;i<8;i++)
		{
		a[i].getdata();
		}
		for(i=0;i<8;i++)
		{
			6M
			
			Creating Class 2M
			
			Logic to Display object with given conditio n 1M
			
			Creating 8 objects 1M
			
			Calling function s 2M

		a[i].putdata();
		}
		getch();
		}	

6	**a)**	**Attempt any TWO of the following** **(i) Write a C++ program to find whether the entered number is even or odd.** **(ii) Write a C++ program to declare a structure employee with members as empid and empname. Accept and display data for one employee using structure variable.**	**12** **6M**

	Ans	**(i) Write a C++ program to find whether the entered number is even or odd.**
		
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int num;
		clrscr();
		cout<<"\nEnter a Number ";
		cin>>num;
		if(num%2==0)
		{
		cout<<"\nEntered number is even";
		}
		else
		{
		cout<<"\nEntered number is odd";
		}
		getch();
		}
		
		(ii) Write a C++ program to declare a structure employee with members as empid and empname. Accept and display data for one employee using structure variable.
		
		#include<iostream.h>
		#include<conio.h>
			Accepting Number 1M
			
			Condition to check number 1M
			
			Display result 1M
			
			Creating structur

		struct employee
		{
		int empid;
		char empname[10];
		};
		void main()
		{
		employee e;
		clrscr();
		cout<<"\nEnter employee id and Employee Name ";
		cin>>e.empid>>e.empname;
		cout<<"\mThe Employee Id is "<<e.empid;
		cout<<"\nThe Employee Name is "<<e.empname;
		getch();
		}
			e with specified member 1M
			
			Accepting and displaying values 2M

	b)	**Write a C++ program to implement following inheritance.**  **Accept and display data for one programmer and one manager. Make display function virtual.**	**6M**
	Ans.	#include<iostream.h> #include<conio.h> class Employee { int empid,empcode; public: void emp() { cout<<"\nEnter an employee id "; cin>>empid; cout<<"\nEnter an employee code "; cin>>empcode;	***Creating all classes 3M***

```
        }
void virtual display()
        {
        cout<<"\nEmployee id "<<empid;
        cout<<"\nEmployee code"<<empcode;
        }
 };
class Programmer : public Employee
 {
char Skill[10];
public:
void getskill()
        {
        cout<<"\nEnter a Skill for Programmer ";
        cin>>Skill;
        }
void display()
        {
        cout<<"\nThe Programmer Skill is "<<Skill;
        }
 };
class Manager : public Employee
 {
char department[10];
public:
void getdept()
        {
        cout<<"\nEnter a Department for Manager ";
        cin>>department;
        }
void display()
        {
        cout<<"\nThe Department of Manager is "<<department;
        }
 };
void main()
 {
  Employee e, *eptr;
  Programmer p;
```

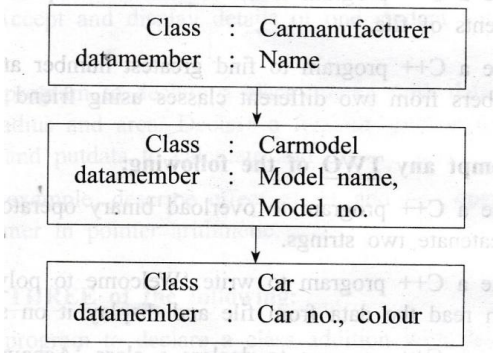
Main Function 3M

```
   Manager m;
clrscr();
cout<<"\nFor Programmer Class ";
eptr = &e;
eptr->emp();
p.getskill();
eptr->display();
eptr= &p;
eptr->display();

cout<<"\nFor Manager Class ";
eptr = &e;
eptr->emp();
m.getdept();
eptr->display();
eptr= &m;
eptr->display();

getch();
  }
```

c) **Write a C++ program for following multilevel inheritance.** **6M**



Accept and display data for one car with all details.

Ans

```
#include<iostream.h>
#include<conio.h>
class Carmanufacturer
  {
char Name[10];
```

Declarat

		public:
		void getcarm()
		{
		cout<<"\nEnter Car Name ";
		cin>>Name;
		}
		
		void putcarm()
		{
		cout<<"\nThe Car Name is "<<Name;
		}
		};
		class Carmodel : public Carmanufacturer
		{
		char Modelname[10];
		int Modelno;
		public:
		void getcarmodel()
		{
		cout<<"\nEnter Car Model Name and Model No. ";
		cin>>Modelname>>Modelno;
		}
		void putcarmodel()
		{
		cout<<"\nEnter Car Model Name and Model No.
		"<<Modelname<<" "<<Modelno;
		}
		};
		class Car: public Carmodel
		{
		char colour[10], Carno[10];
		public:
		void getcar()
		{
		cout<<"\nEnter Car colour and car number";
		cin>>colour>>Carno;
		}
		void putcar()
		{
			ion & Definitio n of all classes 3M

		cout<<"\nEnter Car colour and car number "<<colour<<" "<<Carno;
		}
		};
		void main()
		{
		Car c;
		clrscr();
		c.getcarm();
		c.getcarmodel();
		c.getcar();
		c.putcarm();
		c.putcarmodel();
		c.putcar();
		getch();
		}
			Main function 3M

Boost your Diploma Exams Preparation

Msbte Diploma Exam Papers

GET IT ON Google Play

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.		**Attempt any FIVEof the following:**	**10**
	a) Ans.	**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.	**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.		**Attempt any <u>THREE</u> of the following:**	**12**
	a)	**Write a 'C++' program to find factorial of given number using loop.** ***(Note: Any other correct logic shall be considered)***	**4M**

	Ans.	#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();
		}
			Correct logic 2M
			Correct syntax 2M

	b)	**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)***	**4M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class COLLEGE
		{
		protected:
		int collegecode;
		};
			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**

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.

Any four differences 1M each

		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)
	
	Ans.
		Accept data for five students and display it. Write a C++ program to displya sum of array elements of array size n.
		(Note: Any other correct logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int arr[20],i,n,sum=0;
			4M

```
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) Describe with examples, passing parameters to base class constructor and derived class constructor by creating object of derived class. **4M**

Ans.

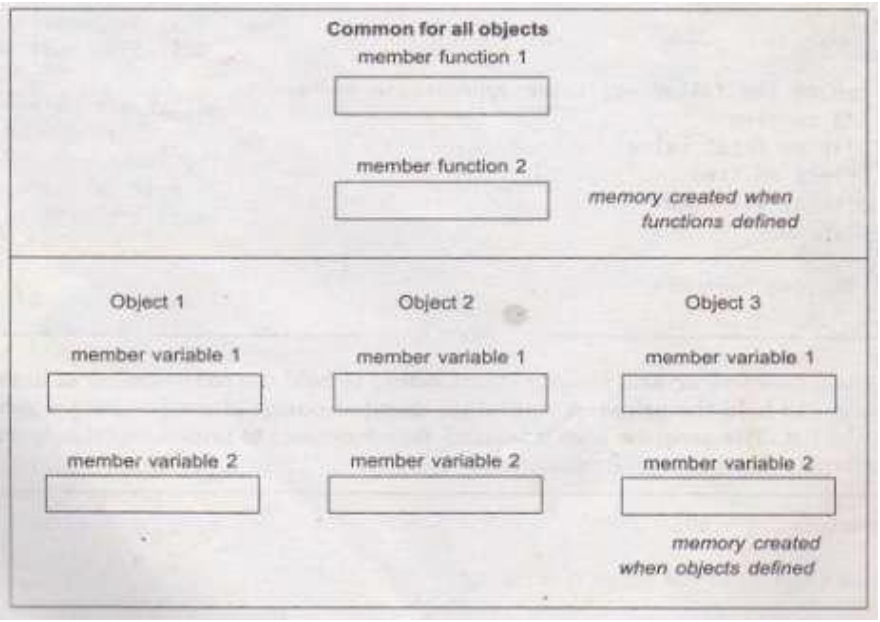
When a class is declared, a constructor can be declared inside the class to initialize data members. When a base class contains a constructor with one or more arguments then it is mandatory for the derived class to have a constructor and pass arguments to the base class constructor. When both the derived 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.

Correct Description 2M

General form to declare derived class constructor:

```
Derived-constructor (arglist1, arglist (D)):Base1(arglist1)
{
Body of derived class constructor
}
```

		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();
		}
		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.
			Correct example 2M

	d) **Ans.**	**Describe how memory is allocated to objects of class with suitable diagram.** **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:  **Fig: Memory allocation for objects**	**4M** ***Correct description 2M*** ***Correct diagram for memory allocation of objects 2M***
4.	**a)**	**Attempt any <u>THREE</u> of the following:** **Write a program to implement multiple inheritance as shown in following Figure No.1:**	**12** **4M**

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

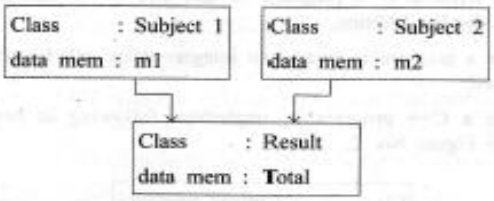


Fig. No. 1

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()
 {
```

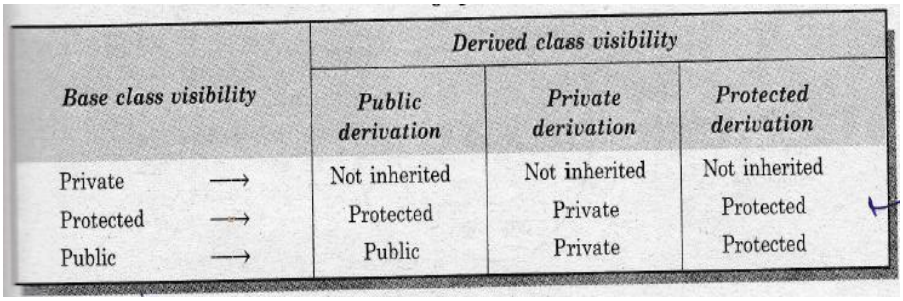
Definition of Class Subject1 1M

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) **Ans.**	**State and describe visibility modes and its effects used in inheritance.** ***(Note: Diagram is optional)*** Different visibility modes are: 1. Private 2. Protected 3. Public Effects of visibility modes in inheritance:  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.	**4M** ***State visibility modes 1M*** ***Description of effect of visibility mode in inheritance 1M each***
	d) **Ans.**	**Write a C++ program to count number of spaces in text file.** ***(Note: Any other correct logic shall be considered)*** **Program:** #include<iostream.h> #include<conio.h> #include<fstream.h> void main() {	**4M**

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

```
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) Ans. **Differentiate between contractor and destructor.**
(*Note: Contractor shall be considered as Constructor.)*

4M

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)
	
	Ans.
		Write a C++ program to append data from abc.txt to xyz.txt file.
		(Note: Any other correct logic shall be considered)
		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;
		}
			6M
			
			Correct logic 3M
			
			Correct Syntax 3M

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

		Output: Hello World	

	c)
	
	Ans.
		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.
		(Note: Any other relevant logic should be considered).
		#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();
		}
			6M
			
			Class student 1M
			
			Construc tor definitio n with default value 2M
			
			Display function definitio n 1M
			
			Main function definitio 2M

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

6.	**a)**	**Attempt any TWO of the following:** **(i) Describe structure of C++ program with diagram.** **(ii) Write a C++ program to add two 3 x 3 matrices and display addition.**	**12** **6M**
	Ans.	**(i) Describe structure of C++ program with diagram.** INCLUDE HEADER FILES DECLARE CLASS DEFINE MEMBER FUNCTIONS DEFINE MAIN FUNCTION **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. **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.	***Correct diagram 1M*** ***Description 2M***

	Ans.	**(ii) Write a C++ program to add two 3 x 3 matrices and display addition.**
		(Note: Any other relevant logic should be considered).
		#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

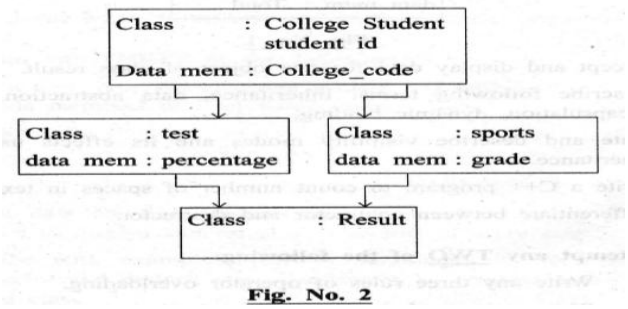
Displaying addition 1M

SUMMER – 2019 EXAMINATION
MODEL ANSWER

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

	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:**  **Fig. No. 2** **Accept and display data for one object of class result (Hint: use virtual base class).**	**6M**

	Ans.	***(Note: Any other relevant logic should be considered).***
		# 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) definitio n 1M
			
			Use of virtual base class 1M
			
			Main function definitio n 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();
}
```

Boost your Diploma Exams Preparation

Msbte Diploma Exam Papers

GET IT ON Google Play

WINTER – 2019 EXAMINATION
MODEL ANSWER

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

Important Instructions to examiners:

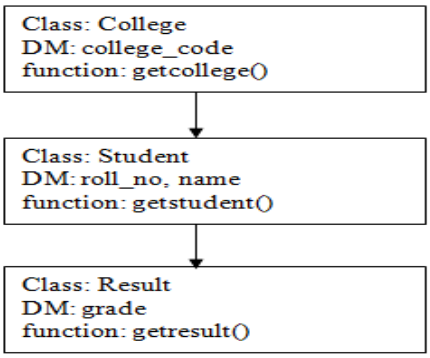
1) The answers should be examined by key words and not as word-to-word as given in the model answer scheme.
2) The model answer and the answer written by candidate may vary but the examiner may try to assess the understanding level of the candidate.
3) The language errors such as grammatical, spelling 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

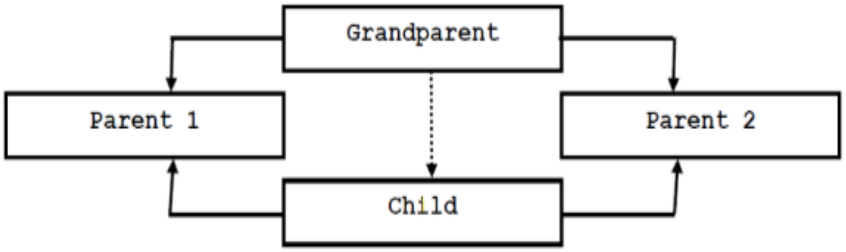
Any two differences 1M each</td></tr>
</table>

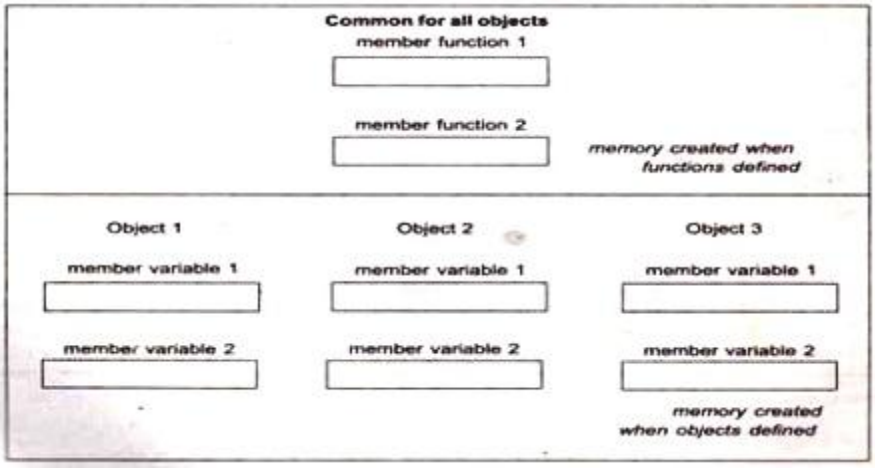
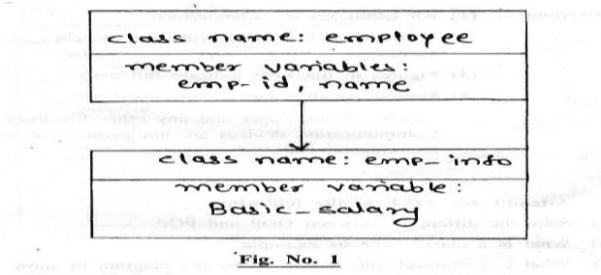
		6 | Object oriented approach is used in C++ language. | Procedure oriented approach is used in C language.	

	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

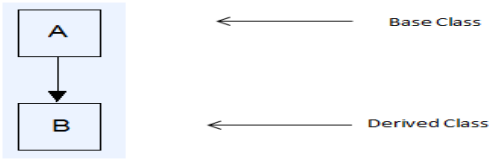
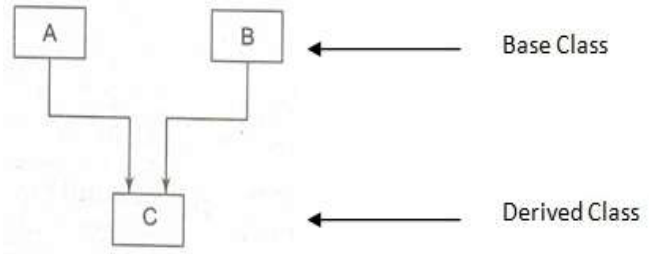
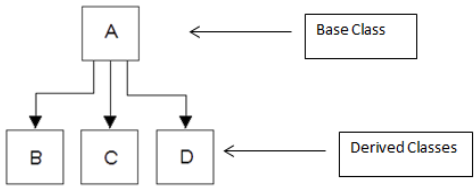
	c) Ans.	**Write any four benefits of OOP.** **Benefits of OOP:** 1. We can eliminate redundant code and extend the use of existing classes. 2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads 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***

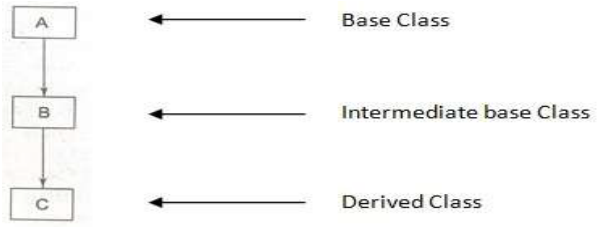
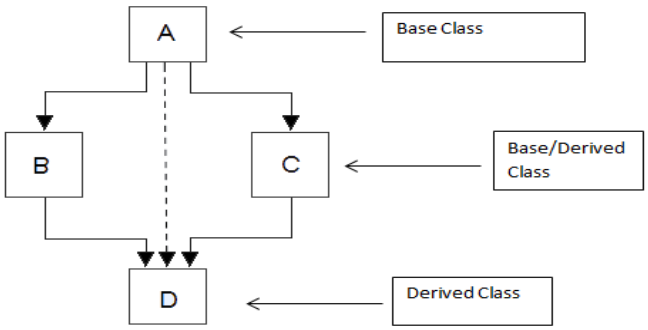
		program. 6. It is easy to partition the work in a project based on objects. 7. The data-centered design approach enables us to capture more details of a model in implementable form. 8. Object-oriented systems can be easily upgraded from small to large systems. 9. Message passing techniques for communication 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**

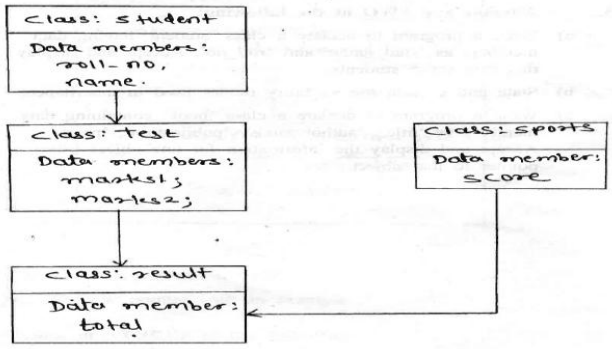


Fig. No. 2

	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