MODEL ANSWER

SUMMER - 2017 EXAMINATION

Subject: Object Oriented Programming **Subject Code:** **17432**

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)**	**Attempt any SIX of the following:**	**12**
	(a) Ans.	**Which are the input-output operator in C++? Give suitable example.** 1. Input operator: >> extraction or get from operator *Example*: cin>> number; 2. Output operator: << insertion or put to operator *Example*: cout<<number;	**2M** ***List of two operators-1M*** ***Example of each ½ M***
	(b) Ans.	**Give significance of '&' and '*' operators.** **Address operator:-&** It is used to retrieve address of a variable. With address operator address of a variable can be stored in pointer variable. **Pointer operator:- * operator** It is used to declare a pointer variable. Also used as 'value at' operator to read value stored inside the address pointed by pointer.	**2M** ***Significance of each 1M***

	(c)	**Calculate the size of object B1 defined in following class:** **Class Book** **{** **char B_name [15];** **int B_id;** **int price;** **};** **Book B1;**	**2M** ***Correct answer2 M***
	Ans.	19 bytes	
	(d)	**List any four types of constructor.**	**2M**
	Ans.	1) Default constructor 2) Parameterized constructor 3) Copy Constructor 4) Constructor with default value 5) Multiple constructor/overloaded constructor	***Any four types ½ M each***
	(e)	**What is polymorphism? List its types.**	**2M**
	Ans.	**Polymorphism-** It is the ability to take more than one form. An operation may exhibit different behaviors in different instances. **Types –** 1) Compile time polymorphism 2) Run time polymorphism	***Definition 1M Two types ½M each***
	(f)	**Define derived class. Give one example.**	**2M**

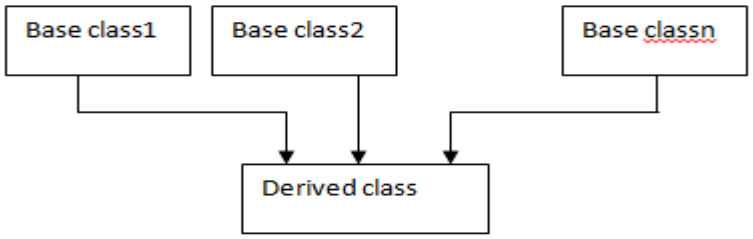
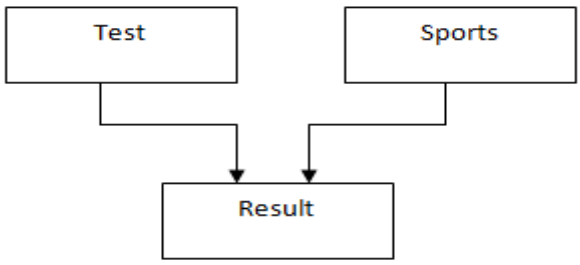
	Ans.	**Derived class**:- a new class derived from an old class is called as derived class.
		Example:-
		class A
		{
		};
		class B: public A
		{
		};
		In the above example class B is a derived class.
			Definition 1M
			
			Example 1M

	(g)	**Write syntax to create a pointer for object.**	**2M**
	Ans.	**Syntax:-** class_name *pointer_name,object_name; pointer_name=&object_name;	***Correct syntax 2M***

	(h)
	
	Ans.
		Write syntax for overloaded constructor.
		(Note: Any relevant syntax shall be considered).
		syntax:-
		class class_name
		{
		public:
		class_name() //constructor name is same as class name
		{
		}
		class_name(argument)//constructor name is same as class name
		{
		}
		};
			2M
			
			Correct syntax 2M

1.	**(B)**
	(a)
	
	Ans.
		Attempt any TWO of the following:
		What do you mean by default argument? Illustrate concept of constructor with default argument using suitable example.
		Default argument:-
		Initializing an argument with a value while defining a constructor is referred as constructor with default value.
		
		When a constructor with default value is declared in a class, it does not require object to pass value for default argument. Constructor will execute without passing default argument value with the object. If object contains value for default argument, then passed value overwrites the default value.
		
		Example:-
		class ABC
		{
		intx,y;
		public:
		ABC(intp,int q=10)
		{
		x=p;
		y=q;
		}
		};
		void main()
		{
			8
			4M
			
			Meaning of default argument 1M
			
			Example 2M

		ABC obj1(5); ABC obj2(20,30); } In above example, obj1 passes one argument to constructor function so x will have value as 5 and y will have value as default value 10. Obj2 passes two arguments to constructor function so x will have value as 20 and y will have value as 30. In obj2, default value is overwritten with passed value.	***Explanation1M***

	(b)
	Ans.
		Draw and explain multiple inheritance with suitable example.
		Multiple Inheritance: When a single class is derived from more than one base classes, it is referred as multiple inheritance.
		Syntax:
		 
		
		Example:-
		 
		
		In the above example class 'result' is a single derived class derived from two base classes base class 'test' and base class 'sports'.
		class Test
		{
		};
		class Sports
		{
		};
		class Result:publicTest,public Sports
		{
			4M
			
			Explanation 2M
			
			Diagram /example 2M

		}; **What are the rules governing the declaration of destructor member function?** **Rules for declaration of destructor member function:** 1. Destructor name is same as class name but is preceded by a tilde. 2. Destructor is declared in public area of a class. 3. Destructor never takes any argument. 4. Destructor never returns any value.	**4M** ***Any four rules 1M each***

2.	**(a)** Ans.	**Attempt any FOUR of the following:**
		Explain access specifiers with suitable example.
		Access specifiers :
		1. private
		2. protected
		3. public
		
		Private access specifier: Class members declared as private can be accessed only from within the class. Outside class access is not allowed for private members of class. By default members are private.
		
		Protected access specifier: Class members declared as protected can be accessed by the member functions within its class and any class immediately derived from it. These members cannot be accessed by the functions outside these two classes.
		
		Public access specifier: Class members declared as public can be accessed from outside the class also.
		
		Example:-
		class base
		{
		private:
		int a;
		protected:
		int b;
		public:
		void display()
		{
		cout<<a<<b;
		}
			16
			4M
			Three access specifiers 1M each
			Example 1M

		};
		class derived:public base
		{
		public:
		void show()
		{
		cout<<b;
		}
		};
		void main()
		{
		derived d;
		d.display();
		d.show();
		}
		
		In the above example, variable 'a' can be access by its member function 'display ()' as it is a private variable. Variable 'b' can be accessed by its member function 'display ()' as well as member function 'show ()' of its derived class as it is a protected member. Member function 'display ()' and 'show ()' can be accessed from main () as they are public members of class.	

	(b) Ans.	**What is virtual function? Why we need virtual function?** ***(Note- Program/example is optional)*** **Definition:** A virtual function is a member function that is declared within a base class and redefined by its derived class. When base class and its derived class both contain same name and prototype member function then derived class function overrides base class function. Base class pointer is used to refer member functions of its class as well as its derived class. When base pointer is used to refer to functions, it ignores the contents of the pointer and selects the member function that matches the function call. When both the classes contain same name and prototype function, base pointer executes a function from base class without considering the address inside the pointer. To execute derived class version of the overridden function virtual keyword is used with base class function. When a function is made virtual, compiler checks the address stored inside the pointer. If the pointer points to base class then function from base class gets	**4M** ***Definitio n 1M*** ***Explana tion 3M***

		executed. If it contains address of derived class then function from derived class gets executed.
		
		Run time polymorphism requires virtual function to execute same name function from base class and derived class depending on address stored inside the pointer.
		
		Program/Example:
		#include<iostream.h>
		class Base
		{
		public:
		virtual void show( )
		{
		cout<<"\nshow base";
		}
		};
		class Derived : public Base
		{
		public:
		void show( )
		{
		cout<<"\nshow derived";
		}
		};
		void main( )
		{
		base B;
		derived D;
		base *bptr;
		bptr=&B
		bptr-> show( );
		bptr=&D;
		bptr-> show( );
		}	

	(c) Ans.	**Write a program that illustrate multilevel inheritance.** ***(Note: Any program showing multilevel inheritance shall be considered).*** #include<iostream.h>	**4M**

		#include<conio.h>
		class student
		{
		protected:
		int roll_no;
		char name[10];
		public:
		void getstudent()
		{
		cout<<"enter roll number and name";
		cin>>roll_no>>name;
		}
		void putstudent()
		{
		cout<<roll_no<<name;
		}
		};
		class test:public student
		{
		protected:
		int marks1,marks2;
		public:
		void gettest()
		{
		cout<<"enter marks";
		cin>>marks1>>marks2;
		}
		void puttest()
		{
		cout<<"marks1="<<marks1<<"marks2="<<marks2;
		}
		};
		class result:public test
		{
		int total;
		public:
		void display()
		{
		total=marks1+marks2;
			Program with correct logic 2M
			
			Correct syntax 2M

```
putstudent();
puttest();
cout<<"total="<<total;
}
};
void main()
{
result r;
clrscr();
r.getstudent();
r.gettest();
r.display();
getch();
}
```

(d) Ans.

Differentiate between POP and OOP.(4 points) **4M**

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 four differences 1M each

(e) Ans.

What is static member function? How is it declare? **4M**

A member function that can access to only other static members declared in the same class is known as static member function.

A static member function can be called using the class name instead of its object.

Definition 2M

		class_name**::**function_name;
		Syntax for declaration:
		
		static return_typefunction_name()
		{
		functiona body
		}
		
		Example:-
		static void showcount()
		{
		cout<<count;
		}
			Declaration 2M

	(f)
	
	Ans.
		Write a program to declare class Account having data member as acc_no and balance. Accept and display data for five object using pointer to array of object.
		#include<iostream.h>
		#include<conio.h>
		class account
		{
		int acc_no,balance;
		public:
		void accept()
		{
		cin>>acc_no>>balance;
		}
		void display()
		{
		cout<<acc_no<<endl<<balance<<endl;
		}
		};
		void main()
		{
		int i;
		account *ptr=new account[5];
		account *ptr1=ptr;
		clrscr();
		for(i=0;i<5;i++)
		{
			4M
			
			Class declaration & definition 2M
			
			main() showing use of pointer 2M

		ptr1->accept();
		ptr1++;
		}
		ptr1=ptr;
		for(i=0;i<5;i++)
		{
		ptr1->display();
		ptr1++;
		}
		getch();
		}	

3.	**(a)** Ans.	**Attempt any FOUR of the following:** **Explain the structure of C++ program with suitable example.** General C++ program has following structure. 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. ***Example:*** #include<iostream.h. #include<conio.h>	**16** **4M** ***Structure 1M*** ***Description 2M***

		class example
		{
		int roll;
		char name[10];
		public:
		void accept( )
		{
		cout<<"Enter Marks for subject 1 and Subject 2";
		cin>>roll>>name;
		}
		void display( )
		{
		cout<<"Roll Number is "<<roll;
		cout<<"\n Name is "<<name;
		}
		};
		void main( )
		{
		example d;
		clrscr( );
		d.accept( );
		d.display( );
		getch( );
		}
			Example 1M

	(b)
	Ans.
		Explain multiple constructor in class. Give suitable example.
		Multiple constructor in which a class can contain more than one constructor. This is known as constructor overloading. All constructor are defined with the same name as the class they belong to. All the constructors contain different number of arguments. Depending upon the number of arguments, the compiler executes appropriate constructor.
		Multiple constructor can be declared in different ways:
		integer(); // No arguments
		integer(int, int); // Two arguments
		When the object is created the first constructor invoked.
		In the first case, the constructor itself supplies the data values and no values are passed by the calling program.
		In the second case, the function call passes the appropriate values from main ( ).
			4M
			Explanation 2M

C++ permits us to use both these constructors in the same class.
For example, we could define a class as follows:

Program:

```
#include<iostream.h>
#include<conio.h>
class integer
{
int m, n;
public:
integer()
{
        m = 0;
        n = 0;
}               // constructor 1
integer(int a, int b)
{
        m = a;
        n = b;
        cout<<"value of m="<<a;
        cout<<"value of n="<<b;
}               // constructor 2
};
void main()
{
        clrscr();
        integer i1;
        integer i2(20,40);
        getch();
}
```

Example 2M

This declared three constructors for an integer object. The first constructor receives no arguments, the second receives two integer arguments and the third receives one integer object as an argument. For example, the declaration.

integer i1;

would automatically invoke the first constructor and set both m and n of i1 to zero. The statement

integer i2 (20, 40);

		would call the second constructor which will initialize the data members m and n i2 to 20 and 40 respectively. So the process of sharing the same name by two or more functions is referred to as function overloading.	

	(c) Ans.	**Explain abstract class with suitable example.**
		An abstract class is designed to act as base class. It is not used to create objects.
		An abstract class is used to define an implementation and is intended to be inherited from by concrete classes. An abstract class is a class that is designed to be specifically used as a base class.
		
		#include<iostream.h>
		class employee
		{
		protected:
		int emp_no;
		public:
		void getdata()
		{
		cout<<"\n Enter employee no.";
		cin>>emp_no;
		}
		void display()
		{
		cout<<"\n Employee no. is :"<<emp_no;
		}
		};
		class fitness:public employee
		{
		protected:
		float height;
		public:
		void getdata()
		{
		employee::getdata();
		cout<<"\n Enter height:";
		cin>>height;
		}
		void display()
		{
			4M
			
			Explanation 2M
			
			Example 2M

```
                                        employee::display();
                                        cout<<"\nheight is :"<<height;
                                }
};
void main()
{
        fitness f;
        f.getdata();
        f.display();
}
```

In the above example, class employee is an abstract class since its object is not created in main().Its members are accessed with the object of its derived class.

(d) **Write a program using function overloading to swap two integer number and swap two float number.** **4M**

Ans.

```
#include<iostream.h>
#include<conio.h>
int swap(int a,int b);
float swap(float c, float d);
int swap(int a,int b)
{
        int temp;
        temp=a;
        a=b;
        b=temp;
        cout<<a<<","<<b<<endl;
}
float swap(float c, float d)
{
        float temp;
        temp=c;
        c=d;
        d=temp;
        cout<<c<<","<<d<<endl;
}
void main()
{
        clrscr();
```

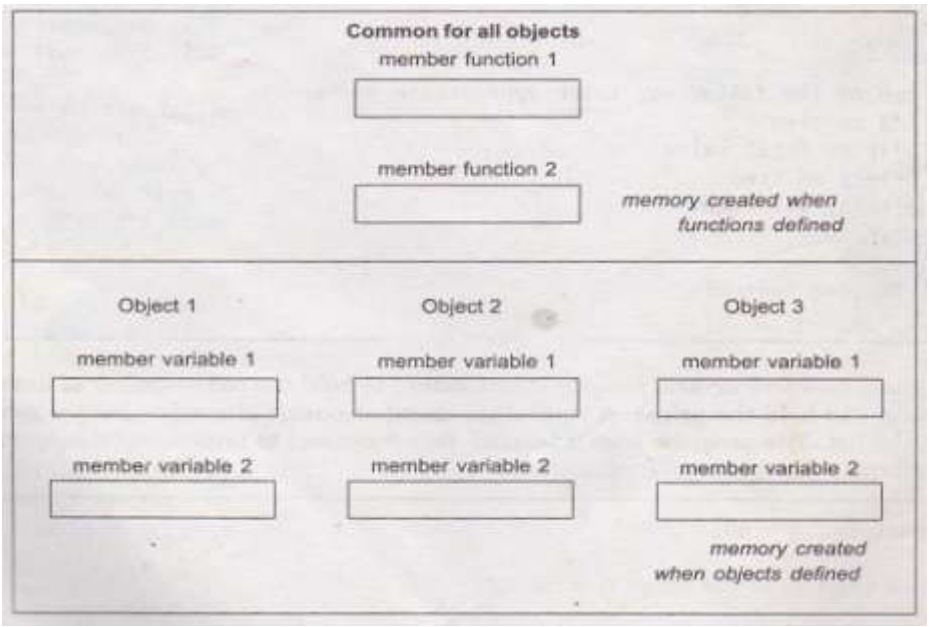
Correct logic 2M

Correct syntax 2M

		int a,b,temp;
		float c,d;
		cout<<"Enter value for a & b="<<endl;
		cin>>a>>b;
		swap(a,b);
		cout<<"Enter value for c & d="<<endl;
		cin>>c>>d;
		swap(c,d);
		getch();
		}	

	(e)
	Ans.
		What is 'this' pointer? Give suitable example.
		'this' pointer:
		1. C++ uses a unique keyword called 'this' to represent an object that invokes a member function.
		2. This unique pointer is automatically passed to a member function when it is invoked.
		3. 'this' is a pointer that always point to the object for which the member function was called.
		4. For example, the function call A.max () will set the pointer 'this' to the address of the object A. Next time suppose we call B.max(), the pointer 'this' will store address of object B.
		Consider the following example:
		#include<conio.h>
		#include<iostream>
		class sample
		{
		int a;
		public:
		void setdata(int x)
		{
		this ->a=x;
		}
		void putdata()
		{
		cout<<this ->a;
		}
		};
		void main()
			4M
			Explanation 2M
			Example 2M

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

	(f) Ans.	**How memory is allocated when multiple object of class are created? Explain with example.** The memory space for object is allocated when they are declared & not when the class is specified. Actually, the member functions are created & 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:  	**4M** ***Explanation 2M*** ***Example 2M***

MODEL ANSWER

SUMMER - 2017 EXAMINATION

Subject: Object Oriented Programming **Subject Code: 17432**

4.		**Attempt any FOUR of the flowing:**	**16**
	(a)	**Write a program to implement single inheritance. Declare base class 'Employee' with emp_no and emp_name. Declare derived class 'Fitness' with height and weight. Accept and display data for one employee.**	**4M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class employee
		{
		protected:
		int emp_no;
		char emp_name[25];
		void getdata()
		{
		cout<<"\n Enter employee no.";
		cin>>emp_no;
		cout<<"\n Enter emplyee name";
		cin>>emp_name;
		}
		void display()
		{
		cout<<"\n Employee no. is :"<<emp_no;
		cout<<"\n Employee name is:"<<emp_name;
		}
		};
		class fitness:public employee
		{
		protected:
		float height,weight;
		public:
		void getdata()
		{
		employee::getdata();
		cout<<"\n Enter height:";
		cin>>height;
		cout<<"\n Enter weight:";
		cin>>weight;
		}
		void display()
			Correct logic 2M
			
			Correct syntax 2M

		{
		employee::display();
		cout<<"\n height is :"<<height;
		cout<<"\n weight is :"<<weight;
		}
		};
		void main()
		{
		clrscr();
		fitness f;
		f.getdata();
		f.display();
		getch();
		}	

	(b)
	
	Ans.
		What is copy constructor? Give the syntax and example for copy constructor.
		The **copy constructor** is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to:
		• Initialize one object from another of the same type.
		• Copy an object to pass it as an argument to a function.
		• Copy an object to return it from a function.
		If a copy constructor is not defined in a class, the compiler itself defines one. If the class has pointer variables and has some dynamic memory allocations, then it is a must to have a copy constructor.
		
		Syntax:
		constructor_name(class_name(data type) &object_name)
		{
		body of copy construtor
		}
		
		Example:
		#include<iostream.h>
		class Point
		{
		private:
		int x, y;
		public:
			4M
			
			Explanation 2M
			
			Syntax 1M
			
			Example

		point(int x1, int y1)
		{
		x = x1;
		y = y1;
		}
		// copy constructor
		point(point &p2)
		{
		x = p2.x;
		y = p2.y;
		}
		intgetX()
		{
		return x;
		}
		intgetY()
		{
		return y;
		}
		};
		int main()
		{
		point p1(10, 15); // Normal constructor is called here
		point p2 = p1; // Copy constructor is called here
		// Let us access values assigned by constructors
		cout<< "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
		cout<< "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();
		return 0;
		}
			1M

	(c) Ans.	**Explain scope resolution operator and memory management operator in C++.** **(i)Scope resolution operator:** In C, the global version of a variable cannot be accessed from within the inner block. C++ resolves this problem by introducing a new operator:: called scope resolution operator. This can be used to uncover a hidden variable. It takes the following form: :: variable; This operator allows access to the global version of a variable.	**4M** ***Operator Explanation 2M each***

		(ii) Memory management operator There are two types of memory management operators in C++: new delete These two memory management operators are used for allocating and freeing memory block in efficient and convenient ways. **New operator:** The new operator in C++ is used for dynamic storage allocation. This operator can be used to create object of any type. **Delete operator:** The delete operator in C++ is used for releasing memory space when the object is no longer needed. Once a new operator is used, it is efficient to use the corresponding delete operator for release of memory.	

	(d)
	Ans.
		Explain friend function with suitable example.
		Friend functions Private and protected members of a class cannot be accessed from outside the same class in which they are declared. However, this rule does not affect friends. A function that is not a member of a class but has access to the class's private and protected members. They are normal external functions that are given special access privileges. Friend function is declared by the class that is granting access. The friend declaration can be placed anywhere in the class declaration. It is not affected by the access control keywords (public, private and protected).
		
		Example:
		#include <iostream.h>
		#include<conio.h>
		class abc
		{
		int a;
		public:
		void get1()
		{
		cin>>a;
		}
		friend void add(abc,xyz);
			4M
			
			Explanation 2M
			
			Example 2M

```
};
class xyz
{
int a;
public:
void get1()
{
cin>>a;
}
friend void add(abc,xyz);
};
void add(abc a1,xyz x1)
{
cout<<a1.a+x1.a;
}
void main()
{
abc a1;
xyz x1;
a1.get1();
x1.get1();
add(a1,x1);
}
```

(e) **Write a program to implement inheritance as shown in fig 1. Assume suitable member function.** **4M**

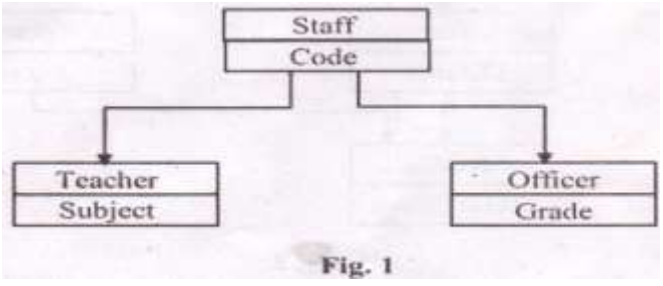


Fig. 1

Ans.

```
#include<iostream.h>
#include<conio.h>
class staff
{
                int code;
public:
```

Correct

```
                    void accept()
                    {
                        cout<<"enter code of staff:"<<endl;
                        cin>>code;
                    }
                    void dis()
                    {
                        cout<<"code="<<code<<endl;
                    }
};
class teacher : public staff
{
    protected:
        char subject[10];
    public:
    void acc1()
    {
        cout<<"enter subject:"<<endl;
        cin>>subject;
    }
    void dis1()
    {
        cout<<"subject="<<subject<<endl;
    }
};
class officer :public staff
{
    protected:
        char grade[5];
    public:
        void acc2()
        {
            cout<<"Enter Grade:"<<endl;
            cin>>grade;
        }
        void dis2()
        {
            cout<<"grade="<<grade<<endl;
        }
```

logic 2M

Correct syntax 2M

		};
		void main()
		{
		clrscr();
		teacher t;
		officer o;
		t.accept();
		t.dis();
		t.acc1();
		t.dis1();
		o.accept();
		o.dis();
		o.acc2();
		o.dis2();
		getch();
		}	

	(f)
	Ans.
		Write a program to insert an element at location of array.
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int a[5],a2[6],i,*a1,no,loc;
		clrscr();
		a1=&a[0];
		cout<<"\n\t Enter array elements:\n";
		for(i=0;i<5;i++)
		{
		cout<<"\n\t Enter "<<i <<" element:";
		cin>>*a1;
		a1++;
		}
		cout<<"\n\t\t Enter element to be inserted at what location::";
		cin>>no>>loc;
		a1=&a[0];
		for(i=0;i<loc;i++)
		{
		a2[i]=*a1;
		cout<<" a2["<<i<<"]= "<<a2[i];
		a1++;
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

```
        }
        a2[loc]=no;
        a1=&a[loc];
        for(i=loc+1;i<=5;i++)
        {
                a2[i]=*a1;
                a1++;
        }
        a1=&a2[0];
        cout<<"\n\t\t New Array is:\n";
        for(i=0;i<6;i++)
        {
        cout<<"\n\t\t Element "<<i<<" :: "<<*a1;
        a1++;
        }
        getch();
}
```

5.	**(a)** Ans.	**Attempt any FOUR of the following:** **State any four rules for 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 explicit argument and those which are overloaded through a friend function take two explicit arguments.	**16** **4M** ***Any four rules 1M each***

		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 return a value. They must not attempt to change their own arguments.	
	(b) Ans.	**Give syntax and example of defining structure and declaring structures variables.** **Definition:-** Structure is a collection of different data types written under a common name. It is a user defined data type. ***Syntax:*** struct structure_name { data_type variable 1; data_typevariable 2; . . . data_type variable n; }structure_variable1,…,structure_variable n; **OR** Structure variable can be created inside main() struct structure_name structure_variable; ***Example:*** struct book { char book_name[10]; char Author_name[10]; float price; }b; **OR** void main() { struct book b; . .	**4M** ***Syntax 2M*** ***Example 2M***

		}	
	(c)	**Create class shape. Derive two classes Triangle and Rectangle. Accept dimensions of Triangle and Rectangle with appropriate functions. Make area () function virtual which is common to all classes. With area function calculate area of triangle and rectangle. Display the result.**	**4M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class shape
		{
		public:
		int l, b, h;
		virtual void area()=0;
		};
		class triangle:public shape
		{
		public:
		void getdata()
		{
		cout<<"\n enter dimensions of triangle:\n length:";
		cin>>l;
		cout<<"\n height:";
		cin>>h;
		}
		void area()
		{
		int a=0.5*l*h;
		cout<<"\n area of triangle:"<<a;
		}
		};
		class rectangle:public shape
		{
		public:
		void getdata()
		{
		cout<<"\n enter dimensions of rectangle:\n length:";
		cin>>l;
		cout<<"\n breadth:";
		cin>>b;
			Correct logic 2M
			
			Correct syntax 2M

		}
		void area()
		{
		int a=l*b;
		cout<<"area of rectangle:"<<a;
		}
		};
		void main()
		{
		clrscr();
		shape *p;
		triangle t;
		rectangle r;
		p=&t;
		t.getdata();
		p->area();
		p=&r;
		r.getdata();
		p->area();
		getch()
		}	

	(d) Ans.	**What are the features of procedure oriented programming?** **Features of procedure oriented programming:** 1.More emphasis is given in doing things. 2.Large program are divided into small modules class functions. 3.Most of functions show global data. 4.Does not support Abstraction, Inheritance, Encapsulation and polymorphism. 5.Employs top –down approach 6.Data moves openly around stem to system from one function to another. 7.New data and functions cannot be easily added whenever required	**4M** ***Any four points each 1M***
	(e) Ans.	**Write a program to search a character in a string using pointer.** #include<iostream.h> #include<conio.h> #include<string.h> void main( ) { char *p, str[15],c;	**4M**

```
int flag=1;
clrscr( );
cout<<'enter string";
cin>>str;
cout<<"/n enter a character to be search";
cin>>c;
while(*p!='\0')
{
    if(c = = *p)
   {
      cout<<"character present">>
      flag = 0;
      break;
    }
    else
    {
       p++;
       flag =1;
     }
}
```

Correct logic 2M

Correct syntax 2M

(f) Ans.

Differentiate between compile time polymorphism and runtime polymorphism. (4 points)

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.

4M

Any four relevant points 1M each

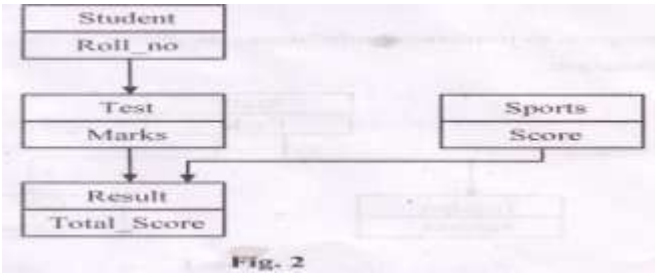
MODEL ANSWER

SUMMER - 2017 EXAMINATION

Subject: Object Oriented Programming **Subject Code:** **17432**

6.		**Attempt any TWO of the following:**	**16**
	(a)	**Write a program declare a class student consisting of data member stud_name and Roll_no. Write program with member function accept( ) to accept and display ( ) to display the data for four students.**	**8M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class student
		{
		private:
		char stud_name[10];
		int roll_no;
		public:
		void accept()
		{
		cout<<"enter student Details":
		cout<<enter name and roll number:";
		cin>>stud_name;
		cin>>roll_no;
		}
		void display()
		{
		cout<<"student name::"<<stud_name<<'\n';
		cout<<"student Roll number::"<<roll_no<<'\n';
		}
		};
		void main()
		{
		student s[4];
		int i;
		for(i=0;i<=3;i++)
		{
		s[i].accept();
		}
		for(i=0;i<=3;i++)
		{
		s[i].display();
		}
		getch();
			Class declaration and definition 4M
			
			Main function with loop 4M

		}	
	(b)	**Write a program to implement inheritance as shown in Fig.2 Assume suitable member function.**  Fig. 2	**8M**

	Ans.	#include<iostream.h>
		#include<conio.h>
		class student
		{
		protected:
		int roll_no;
		public:
		void get_stud()
		{
		cout<<"enter roll number of student";
		cin>>roll_no;
		}
		void disp_stud()
		{
		cout<<" roll number of student"<<roll_no;
		}
		};
		class test:public student
		{
		protected:
		int m1,m2;
		public:
		void get_marks()
		{
		cout<<"enter marks of student";
		cin>>m1>>m2;
		}
			Declaration & definition of class Student 1M
			Sports 1M
			Test 2M
			Result 2M
			Main() 2M

```
void disp_marks()
{
cout<<" marks of student"<<m1<<'\n'<<m2;
}
};
class sports
{
protected:
int score;
public:
void get_score()
{
cout<<"enter score";
cin>>score;
}
void disp_score()
{
cout<<" score"<<score;
}
};
class result:public test,public sports
{
int total_score;
public:
void display()
{
total_score=m1+m2+score;
cout<<"total score="<<total_score;
}
};
void main()
{
result r;
r. get_stud();
r. disp_stud();
r. get_marks();
r, disp_marks();
r. get_score();
r. disp_score();
```

		r. display();
		getch();
		}	

	(c)
	Ans.
		Write a program to display string in reverse order by using pointer.
		#include<iostream.h>
		#include<conio.h>
		#include<string.h>
		void main()
		{
		char str1[10],*ptr;
		int l=0;
		cout<<"enter string:";
		cin>>str1;
		ptr=&str1[0];
		while(*ptr!='\0')
		{
		l++;
		ptr++;
		}
		while(l>0)
		{
		ptr--;
		cout<<*ptr;
		l--;
		}
		getch();
		}
			4M
			Correct logic 4M
			Correct syntax 4M