MODEL ANSWER

SUMMER – 2018 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) (a) Ans.**	**Attempt any SIX of the following:** **State any four object oriented programming language.** **Object oriented programming language:** • C++ • Smalltalk • Object pascal • java • Simula • Ada • Turbo pascal • Eiffel • C# • Python	**12** **2M** ***Any 4 languages ½ M each***
	(b) Ans.	**Define pointer. Give syntax for declaration of pointer.** **Definition:** Pointer is a variable that holds memory address of another variable of similar data type.	**2M** ***Definition 1M***

MODEL ANSWER

SUMMER – 2018 EXAMINATION

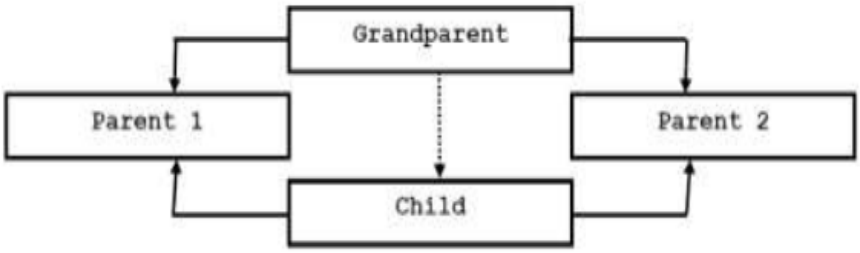
Subject: Object Oriented Programming **Subject Code:** **17432**

		Syntax for declaration: data_type *pointer_variable_name;	***Syntax 1M***
	(c) Ans.	**List any two properties of static member function.** • A static function can have access to only other static members (function or variables) declared in the same class. • A static member function can be called using the class name(instead of its objects) as follows: Class_name**::**function_name;	**2M** ***Two properties 1M each***
	(d) Ans.	**What is the importance of constructor?** A constructor is important to initialize the objects of its class. It is called constructor because it constructs the value of the data members inside object of the class.	**2M** ***Importance of constructor 2M***
	(e) Ans.	**Define polymorphism. List types of polymorphism.** **Definition:-** Polymorphism 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	**2M** ***Correct definition 1M*** ***Types 1M***
	(f) Ans.	**Define abstract class.** An abstract class is a class that is designed only to act as base class. It is not used to create objects.	**2M** ***Definition 2M***
	(g) Ans.	**What is the use of this pointer?** 'this' pointer is used to represent an object that invokes a member function. It points to the object for which the function is called. It is also used to access members of object inside function definition of called function. Example: this->rollno=1;	**2M** ***Correct Use 2M***
	(h) Ans.	**How do we invoke a constructor function?** A constructor is invoked automatically when an object of its class is created. *Example:* class ABC { public: ABC( ) {	**2M** ***Correct explanation 2M***

		}
		};
		void main( )
		{
		ABC obj;
		}
		In the above example, creating 'obj' object automatically invokes constructor 'ABC ( )'.	

1.	**(B)**
	(a)
	
	Ans.
		Attempt any TWO of the following:
		Write a program to declare class having data member as hrs, mins, secs. Write constructor to assign values and destructor to destroy values. Accept & display data for one object.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		class time
		{
		private:
		int hrs, mins,sec;
		public:
		time(int h,int m,int s)
		{
		hrs=h;
		mins=m;
		sec=s;
		}
		~time()
		{
		cout<<"hours deleted";
		cout<<"minutes deleted";
		cout<<"seconds deleted";
		}
		void display()
		{
		cout<<"hours="<<hrs;
		cout<<"Minutes="<<mins;
		cout<<"seconds="<<sec;
		}
		};
			8
			4M
			
			Correct logic for construc tor ()- 1M, destruct ed()- 1M,displ ay -1M, main() - 1M

		void main()
		{
		time t(2,43,56);
		t.display();
		getch();
		}	

	(b) Ans.	**What is virtual base class? Explain with suitable diagram.**
		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.
		
		Diagram:
		 
		
		Consider a hybrid inheritance as shown in the above diagram. The child class has two direct base classes, 'parent1'&'parent2' which themselves have a common base class as 'grandparent'. The child inherits the members of 'grandparent' via two separate paths. All the public & protected members of "grandparent" are inherited into "child" twice, first via 'parent1'& again via 'parent 2'. 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
			4M
			
			Correct Definition 1M
			
			Diagram 1M
			
			Explanation 2M

		{
		};
		class Child: public Parent1,public Parent2
		{
		};	

	(c) Ans.	**Describe multiple constructor by giving example.**
		Multiple constructor:
		Multiple constructors is a category of constructor in which a class can have more than one constructor. This is also known as constructor overloading. All constructors are defined with the same name as the class name they belong to. Each of the constructors contains different number of arguments. Depending upon the number of arguments and their data type, the compiler executes appropriate constructor.
		
		Example:-
		#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);
			4M
			
			Description 2M
			
			Example 2M

getch();
}
In the above example, two constructors are defined and invoked; this is referred as multiple constructors.
The first constructor does not accept any argument and the second accepts two integer arguments.
In void main( ):
integer i1; - This statement invokes first constructor.
integer i2 (20, 40); -This statement invokes second constructor.

2. **(a)** **Attempt any FOUR of the following:** **16**

Give four differences between structure and class. **4M**

(Note: Any other relevant point shall be considered).

Ans.

Structure	Class
1. Structure is a collection of logically related data items which can be of similar type or different type.	1. Class is a way of binding data and functions together in one single unit. It is a collection of data members and member functions.
2. In structure, data is not hidden from external use.	2. Class allows data and functions to be hidden from external use.
3. In Structure, by default all members are public.	3. In Class, by default all members are private.
4. In Structure, structure variable is created.	4. In class object is created.
5. *Syntax:* struct structure_name { Data_typevariable1; Data_type variable2; .. .. }structure_variable;	5. *Syntax:* class class_name { Access specifier: declare data members; declare member functions; }object;

Any 4 points 1M each

(b) **Explain concept of function overriding with example.** **4M**

Ans.

Function Overriding:-

When derived class defines same name function, as defined in its base class then it is called as function overriding. In this a function in

Explanation 2M

		the derived class overrides the inherited function.
		
		Example :
		#include<iostream.h>
		#include<conio.h>
		class Base
		{
		public:
		void Display()
		{
		cout<<"\n Display Base";
		}
		};
		class Derived : public Base
		{
		public:
		void Display()
		{
		cout<<"\n Display Derived";
		}
		};
		void main()
		{
		Derived D;
		D.Display();
		}
		
		In the above example, base class and derived class both contains a same name function 'Display'. The derived class overrides the 'Display' function of base class.
			Example 2M

	(c)	**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.** ***(Note: Any other relevant logic shall be considered)***	**4M**
	Ans.	#include<iostream.h> #include<conio.h> class employee {	***Base class definitio n-1M***

		protected:
		int emp_no;
		char emp_name[10];
		public:
		void gete()
		{
		cout<<"enter employee details";
		cin>>emp_no;
		cin>>emp_name;
		}
		void pute()
		{
		cout<<"employee details are=";
		cout<<emp_no<<'\n';
		cout<<emp_name;
		}
		};
		
		class fitness:public employee
		{
		float height,weight;
		public:
		void getft()
		{
		cout<<"enter height and weight";
		cin>>height>>weight;
		}
		void putft()
		{
		cout<<"height and weight is=";
		cout<<height<<weight;
		}
		};
		void main()
		{
		fitness f;
		f.gete();
		f.pute();
		f.getft();
			Derived class definitio n 2M
			
			Main() definitio n 1M

		f.putft();
		getch();
		}	

	(d) **Ans.**	**List features of procedure oriented programming.** **Features of procedure oriented programming:** 1. Emphasis is on doing things (algorithms). 2. Large program are divided into smaller programs known as functions. 3. Most of the functions share global data. 4. Data moves openly around the system from function to another. 5. Functions transform data from one form to another. 6. Employs top –down approach in program design.	**4M** ***Any 4 points 1M each***
	(e) **Ans.**	**Why friend function is required? Give four characteristics of friend function.** **Friend function:** Private members of a class cannot be accessed from outside the class. A non-member function cannot have an access to the private data of a class. Sometimes, two classes may need to share data in a common function. **To access private data of more than one class in one common function**, friend function is required. The common function is declared as a friend function of all those classes from which the function wants to share data. **Characteristics of friend function:** 1. It is not the scope of the class to which it has been declared as friend. 2. Since it is not in the scope of the class it cannot be called using the object of that class. 3. It can be invoked like a normal function without the help of any object. 4. Unlike member functions, it cannot access the member names directly and has to use an object name and dot membership operator with each member name. 5. It can be declared either in the public or the private part of a class without affecting its meaning. 6. Usually it has the objects as the arguments.	**4M** ***Explanation of Friend function need 2M*** ***Any 4 characteristics ½ M each***
	(f) **Ans.**	**Write a program to swap two integer values by using call by reference.** ***(Note: Any other relevant logic shall be considered)***	**4M**

		#include<iostream.h>
		#include<conio.h>
		void swap(int *a,int *b)
		{
		int c;
		c=*a;
		*a=*b;
		*b=c;
		}
		void main()
		{
		int a,b;
		cout<<"Enter Value Of a:";
		cin>>a;
		cout<<"Enter Value of b:";
		cin>>b;
		cout<<"Before swapping :";
		cout<<"Value of a is "<<a<<"\n;
		cout<<"Value of b is "<<b;
		swap(&a,&b);
		cout<<" After swapping:";
		cout<<" value of a is"<<a<<"\n";
		cout<<"Value of b is"<<b;
		getch();
		}
			Swap function with pointer 2M
			
			Main ()2M

3.	**(a) Ans.**	**Attempt any FOUR of the following:** **Explain data encapsulation and data abstraction.** **Data encapsulation:** The wrapping up of data and function into a single unit (called class) is known as encapsulation. The data is not accessible to the outside world, and only those functions which are wrapped in the class can access it. These functions provide the interface between the object's data and the program. Encapsulation is a mechanism that keeps the data and code safe from external interference and misuse. This insulation of the data from direct access by the program is called data hiding or information hiding.	**16** **4M** ***Explanation of data encapsulation 2M***

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

		Data abstraction: Abstraction refers to the act of representing essential features without including the background details or explanation. Data abstraction is the process of defining a data type, often called abstract data type (ADT), together with the principle of data hiding. Classes use the concept of abstraction. They encapsulate all the essential properties of the object that are to be created. The attributes are called as data members as they hold information. The functions that operate on these data are called as member functions.	***Explanation of data abstraction 2M***

	(b)
	
	Ans.
		Explain the concept of overload constructor in class with example.
		Overloaded constructor:
		When more than one constructor function is defined in a class then it is called as overloaded constructor. All constructors are defined with the same name as the class name they belong to. Each of the constructors contains different number of arguments. Depending upon the number of arguments and their data type, the compiler executes appropriate constructor.
		
		Example:-
		#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
			4M
			
			Explanation of overload constructor 2M
			
			Example 2M

```
}
void main()
{
clrscr();
integer i1;
integer i2(20,40);
getch();
}
```

In the above example, constructor is overloaded by defining two constructors in the same class. Both the definitions are different with respect to number of arguments. The first constructor does not accept any argument and the second accepts two integer arguments.

(c) Ans. **What is inheritance and explain visibility modes in detail.** — **4M**

Inheritance:

The mechanism of deriving new class from an old (existing) class is called as inheritance. With inheritance, one class acquires the properties of objects of other classes.

Definition of inheritance 1M

Following are different visibility modes in C++:-

1. Public 2. Private 3. Protected

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

Explanation of three visibility modes 1M each

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 becomes 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 becomes protected members of derived class and public members of base class becomes public members of derived class.	

	(d)
	
	Ans.
		Write a program for overloading of ++unary operator for inch to feet conversion. 12 inch = 1 feet.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		class abc
		{
		int i,f;
		public:
		abc(int f1,int i1)
		{
		f=f1;
		i=i1;
		}
		
		void operator ++()
		{
		while(i>11)
		{
		f++;
		i=i-12;
		}
		cout<<"Number of feet ="<<f<<"Number of inches:"<<i;
		}
		};
		int main()
		{
		clrscr();
		abc a1(2,49);
		++a1;
		getch();
		return 0;
		}
			4M
			
			Correct logic 2M
			
			Correct syntax 2M

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

(e) **Write a program to copy content of one string to another string using pointer to string.** **4M**

(Note: Any other relevant logic shall be considered)

Ans.

```
#include<iostream.h>
#include<conio.h>
void main()
{
        char str1[10],str2[10],*p1,*p2;
        clrscr();
        cout<<"\n Enter a String";
        cin>>str1;
        p1=&str1[0];
        p2=&str2[0];
        while(*p1!='\0')
        {
                *p2=*p1;
                p1++;
                p2++;
        }
        *p2='\0';
        cout<<"Copied String is "<<str2;
        getch();
}
```

Correct logic 2M

Correct syntax 2M

(f) **Differentiate between call by value and call by reference method.** **4M**

(Note: Any other relevant point shall be considered).

Ans.

Call by Value	Call by reference
In call by value, a copy of actual arguments is passed to respective formal arguments.	In call by reference, the location, that is, the address of actual arguments is passed to formal arguments
Actual arguments will remain safe, they cannot be modified in the called function.	Alteration to actual arguments is possible within called function.
Address of the actual and formal arguments are different	Address of the actual and formal arguments are the same
Changes made inside the function is not reflected in other functions	Changes made in the function is reflected outside also.

Any 4 points 1M each

		Example: swap(a,b); //function call
		void swap(int a,int b)// function definition
		{
		}
		
		Example: swap(&a,&b); //function call
		void swap(int *a,int *b)// function definition
		{
		}	

4.	**(a) Ans.**	**Attempt any FOUR of the following:**
		What is abstract class? Give one example of abstract class.
		An abstract class is a class that is designed only 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 by child classes.
		
		Example:
		#include<iostream.h>
		class base
		{
		protected:
		int a;
		public:
		void getdata()
		{
		cin>>a;
		}
		void display()
		{
		cout<<a;
		}
		};
		class derived: public base
		{
		protected:
		int b;
		public:
		void getdata1()
		{
		getdata();
		cin>>b;
		}
		void display1()
			16
			4M
			
			Definition 2M
			
			Example 2M

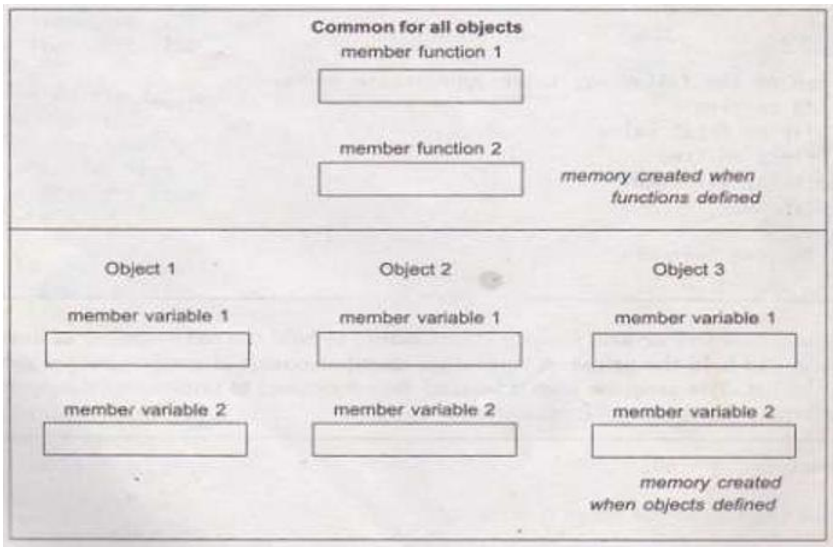
		{
		display();
		cout<<b;
		}
		};
		void main()
		{
		derived d;
		d.getdata1();
		d.display1();
		}
		In the above example, class base is an abstract class since its object is not created in main().Its members are accessed through its derived class.	

	(b)
	
	Ans.
		Write a program to declare class student having data members name and percentage. Write constructor to initialize these data members. Accept and display this data for one object.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		#include<string.h>
		class student
		{
		char name[20];
		float per;
		public:
		student(char n[],float p)
		{
		strcpy(name,n);
		per=p;
		}
		void putdata()
		{
		cout<<"\n\t\t Name ::"<<name;
		cout<<"\n\t\t Per ::"<<per;
		}
		};
		void main()
		{
			4M
			
			Class definition-1M, constructor definition-1M, display-1M,main()-1M

		student S("Sachin",78.00);
		clrscr();
		S.putdata();
		getch();
		}	

	(c)
	Ans.
		Give the syntax and use of following with respect to (i) get() (ii) put().
		(i) get() function:
		The get() function is member of istream class. It is used to read a single character from the keyboard.
		Syntax of get() function:
		get(variable_name);
		
		example:
		char c;
		cin.get(c);
		In the above example, variable 'c' is a character variable. The get () function reads a single character from the keyboard and stores it inside variable 'c'.
		
		(ii) put( ) function:
		The put() function is member of ostream class. It is used to output a single character on the screen.
		Syntax:
		put(character/variable_name);
		example:
		cout.put('X');
		The above example displays character 'X' on the screen.
			4M
			
			syntax of get() and put() 1M each
			
			Use of get() and put() 1M each

	(d) **Ans.**	**Explain the concept of memory allocation for object.** The memory space for object is allocated when it is declared & not when the class is specified. 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 (data) member variables is allocated separately for each object. Separate memory locations for the objects are essential because the (data) member variables will hold different data values for different objects.	**4M** ***Relevant explanation 4M***

		 In the above diagram, member functions 1 and 2 are stored in the common memory space as they require access by all objects. Each object (object 1, object 2, object 3) has its own separate memory space for its member variables.	
	(e) **Ans.**	**Explain the following with syntactic rules: (i) public inheritance (ii) protected inheritance.** **(i) public inheritance:** i) When the visibility-mode is public the base class is publicly inherited. ii) In public inheritance, the public members of the base class become public members of the derived class and therefore they are accessible to the objects of the derived class. iii) When deriving a class from a public base class, protected members of the base class become protected members of the derived class. iv)A base class's private members are never accessible directly from a derived class, but can be accessed through calls to the public and protected members of the base class. v) *Syntax:* class A { Public: Member variables;	**4M** ***Public inheritance 2M***

		Member function;
		};
		class B : public A
		{
		Members of class B
		};
		
		(ii) protected inheritance:
		1. If we want to inherit private data by a class, the only option is to change the visibility limit from private to public, but this will eliminate the advantage of data hiding.
		2. Therefore to achieve data hiding, C++ provides a third visibility modifier, protected which has limited purpose in inheritance.
		3. A member declared protected is accessible by the member functions within its class and any class immediately derived from it. It cannot be accessed by the functions outside these two classes.
		4. In protected inheritance, protected and public members of base class become protected members of derived class.
		5. When a protected member is inherited in public mode, it becomes protected in the derived class too, and therefore is accessible by the member.
		Syntax:
		class A
		{
		protected :
		Member variables;
		Member function;
		};
		class B : protected A
		{
		Members of class B
		};
			Protected inheritance 2M

	(f) Ans.	**Write advantages of pointer.** **The pointer has following advantages:** 1. Pointers reduce the length and complexity of a program. 2. They increase execution speed. 3. Pointer saves the memory. 4. A pointer enables us to access a variable that is defined outside the function.	**4M** ***Any 4 advantages 1M each***

		5. The use of a pointer array of character strings results in saving of data storage space in memory. 6. Pointer supports dynamic memory management.	

5.	**(a)**
	
	Ans.
		Attempt any FOUR of the following:
		Polymorphism is implemented using function overloading. Justify the statement.
		Polymorphism is a mechanism that allows a developer to have more than one function with same name but different signature. In function overloading, one can make use of more than one function with different signature as well. Hence polymorphism is implemented using function overloading, where one can have more than one functions possess same name but different functionality and behavior.
		
		Example:
		#include<iostream.h>
		#include<conio.h>
		int add(int ar[]);
		float add(float arr[]);
		void main()
		{
		int a[10],i,isum;
		float b[5],fsum;
		clrscr();
		cout<<"\nEnter 10 numbers ";
		for(i=0;i<10;i++)
		{
		cin>>a[i];
		}
		cout<<"\nEnter 5 Float numbers ";
		for(i=0;i<5;i++)
		{
		cin>>b[i];
		}
		isum=add(a);
		fsum=add(b);
		cout<<"\nThe addition of 10 integer numbers is "<<isum;
		cout<<"\nThe addition of 05 Float numbers is "<<fsum;
		getch();
			16
			4M
			
			Description 2M
			
			Example 2M

```
  }

int add(int x[])
  {
int sum = 0, i;
for(i=0;i<10;i++)
     {
          sum = sum+x[i];
     }
return sum;
  }

float add(float x[])
  {
float sum = 0, i;
for(i=0;i<5;i++)
     {
          sum = sum+x[i];
     }
return sum;
  }
```

As shown in above example there are two different functions add, having same name but argument and return type differs and performs different tasks. Based on the argument appropriate function will be called. Function selection will be done at compile time itself.

(b) **State characteristic of static data member. Explain why static data member must be defined outside the class.** — **4M**

Ans.

Characteristics of static data members:

1. It is initialized to zero when the first object of its class is created. No other initialization is permitted.
2. Only one copy of that member is created for the entire class.
3. Created copy is shared by all the objects of that class, no matter how many objects are created.
4. It is visible only within the class, but its lifetime is the entire program.

Any 2 Characteristics 2M

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

		Description: Since objects are created anywhere in a program, and all objects refers/shares the value of static member(s) it is necessary to make static members global and re-declared outside of the class.	***Description 2M***
	(c) **Ans.**	**Write rules for overloading operator.** **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. They are sizeof(),membership, pointer-to-member, scope resolution, conditional operators. 6. We can't use friend functions to overload certain operators. However, member functions can 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. 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.	**4M** ***Any four rules 1M each***
	(d) **Ans.**	**Give four differences between object oriented programming and procedure oriented programming.**	**4M**

MODEL ANSWER

SUMMER – 2018 EXAMINATION

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

		<table><tr><th>OBJECT ORIENTED PROGRAMMING (OOP)</th><th>PROCEDURE ORIENTED PROGRAMMING (POP)</th></tr><tr><td>It focuses on data.</td><td>It focuses on procedure.</td></tr><tr><td>Programs are divided into multiple classes and objects.</td><td>Large programs are divided into multiple functions.</td></tr><tr><td>Data is hidden and cannot be accessed by external functions.</td><td>Data move freely around the system from function to function.</td></tr><tr><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>Follows bottom-up approach in program design</td><td>Follows top-down approach in program design.</td></tr></table>	***Any 4 differences 1M each***

	(e)
	
	Ans.
		Write a program to search a number from an array using pointer to array.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		void main()
		{
		int arr[10], key, i, *ptr,flag =0;
		clrscr();
		ptr=&arr[0];
		cout<<"\n Enter 10 numbers";
		for(i=0;i<10;i++)
		{
		cin>>*ptr;
		ptr++;
		}
		ptr=&arr[0];
		cout<<"\n Enter a number to be searched ";
		cin>>key;
		for(i=0;i<10;i++)
		{
		if(*ptr==key)
		{
		flag=1;
			4M
			
			Creating pointer variable 1M
			Accepting value using pointer 1M
			
			Searching element/ number using pointer 2M

		cout<<"\nElement found at position "<<i+1;
		break;
		}
		ptr++;
		}
		if(flag==0)
		{
		cout<<"\nElement does not exist in array ";
		}
		getch();
		}	

	(f) Ans.	**What is the need of virtual function? Explain with example.**
		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 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<<"\n show base";
		}
			4M
			
			Need of virtual function 2M
			
			Example with explanation 2M

		};
		class Derived : public Base
		{
		public:
		void show( )
		{
		cout<<"\n show derived";
		}
		};
		void main( )
		{
		Base B,*bptr;
		Derived D;
		bptr=&B;
		bptr->show( );
		bptr=&D;
		bptr->show( );
		}
		As given in above example, both base and derived class has same function named as show. By creating a pointer object of base class one can invoke desired show function by adjusting pointer position.	

6.		**Attempt any TWO of the following:**	**16**
	(a)	**Write a program to show object as function argument.** ***(Note: Any other relevant logic shall be considered)***	**8M**

	Asn.	#include<iostream.h>
		#include<conio.h>
		#include<string.h>
		class objarg
		{
		char str[10];
		public:
		void get()
		{
		cout<<"\n Enter a Message";
		cin>>str;
		}
		void copy(objarg o)
		{
		strcpy(str,o.str);
			Creating class and object 2M
			
			Function body with object as argument 4M

		}
		void display()
		{
		cout<<"\n Message is "<<str;
		}
		};
		void main()
		{
		objarg o1, o2;
		clrscr();
		o1.get();
		o2.copy(o1);
		o2.display();
		getch();
		}
			Calling function with object as argument 2M

	(b)
	
	Ans.
		Write a program for multiple inheritance.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		class base1
		{
		public:
		void show1()
		{
		cout<<"\nThis is base 1";
		}
		};
		class base2
		{
		public:
		void show2()
		{
		cout<<"\nThis is base 2";
		}
		};
		class derived: public base1, public base2
		{
		public:
		void display()
			8M
			
			Body of more than one base class 2M each
			
			Body of Derived class inheriting properties of more than one base class 2M

		{
		cout<<"\n Executing from derived";
		show1();
		show2();
		}
		};
		void main()
		{
		derived d;
		clrscr();
		d.display();
		getch();
		}
			Calling Using members of Base class in derived class 2M

	(c)
	
	Ans.
		Write a program to find whether the string is palindrome or not.
		(Note: Any other relevant logic shall be considered)
		#include<iostream.h>
		#include<conio.h>
		#include<string.h>
		void main()
		{
		char str1[10],str2[10];
		int c;
		clrscr();
		cout<<"\n Enter string:";
		cin>>str1;
		strcpy(str2,str1);
		strrev(str2);
		cout<<"\n Rverse string :"<<str2;
		c=strcmp(str1,str2);
		if(c==0)
		cout<<"\n String is palindrome";
		else
		cout<<"\n String is not palindrome";
		getch();
		}
		OR
		
		#include<iostream.h>
		#include<conio.h>
			8M
			
			Finding length 2M
			
			Reversal of input string 2M
			
			Identifying palindrome of string via string comparison 4M

```
void main()
  {
char src[10], des[10], *sptr, *dptr;
int len=0;
clrscr();
cout<<"\n Enter a string";
cin>>src;
sptr=&src[0];
while(*sptr!='\0')
     {
          len++;
          sptr++;
     }
cout<<"\n Length of string is "<<len;

sptr--;
dptr=&des[0];
while(len>0)
     {
          *dptr=*sptr;
          sptr--;
          dptr++;
          len--;
     }
   *dptr='\0';
cout<<"\n The Reverse string is "<<des;

sptr=&src[0];
dptr=&des[0];
int flag=0;
while(*dptr != '\0' || *sptr !='\0')
     {
          if(*dptr == *sptr )
            {
          flag =0;
            }
          else
            {
          flag =1;
```

```
            break;
              }
            dptr++;
            sptr++;
        }
if(flag ==0)
        {
            cout<<"\nThe String is Palidrome ";
        }
else
        {
            cout<<"\nString is not palindrome";
        }
getch();
  }
```