B. E.

Fourth Semester Examination, May-2006

OBJECT ORIENTED PROGRAMMING USING C++

Note : Attempt any five questions. All questions carry equal marks.

Q. 1. (a) What do you understand by namespace? Discuss following, using suitable examples :

(a) Nested namespaces

(b) Unnamed namespaces

(c) Namespaces aliases.

Ans. (a) Namespace : It is a new concept introduced by ANSI C++ standard committee. This defines a scope for the identifiers that are used on a program for using the identifiers defined is the namespace, scope, we must include the using directive, like

using namespace std;

Nested Namespace : A member function can be called by using its name inside another member function of the same class, this is known as nesting of member functions.

(b) Unnamed Nemespace : An unnamed namespace is one that does not have a name. Unnamed namespace members occupy global scope and are accessible is all scopes following the declarations in the file. We can access them without using any qualification.

Use of common Unnamed Nespace : A common use of unnamed namespace is to shield global data from potential name classes between files. Every file has its own, unique unnamed namespace.

(c) Namespaces aliases : Namespace namespace-name

```
{
        // declaration of
        // variable, functions,
        classes etc.
}.
```

Q. 2. (a) What do you understand by object oriented programming? What are its advantages over procedure oriented programming? Explain.

Ans. Object Oriented Programming : It is the most recent concept among programming paradigms and still means different things to different people. It is therefore important to have a working diff. of object oriented programming.

We define "object oriented programm." as an approach that provides a way of modularizing programs by creating partitioned mem. area for both data and functions that can be used as template for creating copies of such modules on demand.

Main concepts of object-oriented programming.

- Objects
- Classes
- Data abstraction & encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Message passing

Advantage of object oriented programming over produce prog is :

- Emphasis is on data rather than procedure.
- Program are divided into what are known as objects.
- Data structures are designed such that they characterize the objects.
- Functions that operate on the data of an object are tied together is data structure .
- Data is hidden & cann't accessed by external functions.
- Object may communicate with each other through functions.

Q. 2. (b) What do you understand by access modifiers? Differentiate between publicly and privately inherited classes, using suitable examples.

Ans. The private data of a class can be accessed only through the member functions of that class. The main ( ) cannot statements that access number and cost directly.

The following is format of calling a member function :

Objectname.function name (actual-arguments);

x.getdata (100, 75.5);

By default the member of class is private, if both public & private access modifier are mining, we consider it private by default. It public them if accessible to outside class.

Q. 3. Differentiate between the following :

(a) Class and object

(b) Class and structure

(c) Friend function and friend class.

Ans. (a) Class and object : Class is a way to find the data & its associated functions together. It allows the data (and functions) to be hidden. If necessary, from ext. use. When define a class, we are creating new abstract data type that can treated by other built in data type. Object is an instance of class, which we can used by use of member function of class.

(b) Class and structure : Class def. is similar to structure definition the keyword class specifies that what follow is an abstract data of type class-name the body of class enclosed with in traces & terminated by a semicolon. A structure is a convenient the for handling a group of logically related data items. It is a user defined data type with a template that dives to define its data properties.

(c) Friend function and friend class : Friend function it is not in the scope of class to which it has been declared as friend.

Since it is not in scope of the class, it cannot be called using objects of that class.

It can be invoked liked a normal function without the help of any object e.g.

```
Class ABC
{       public
        .......
        ......
        .....
        friend void xyz (void);
};
```

Friend class : A class can be friend of another class, using keyword Friend use before the class. For e.g. friend class xyz

```
        {
                .....
                .....
                ......
```

Q. 4. What do you understand by operator overloading? Write a program to :

(a) Overload "+" operator to concatenate two strings.

Ans. Operator overloading : The input/output operators << and >> are good example of operator overloading. Although the built in def. of << operator is for shifting of bits.

It is also used for displaying the values of various data types. This has been made possible by the header file iostream where a No. of overloading definitions for << are included. The input/output operators <<and>> are good examples of operator overloading. Although the built in def. of the << operator is for shifting of bits, it is also used for the displaying the values of various data types. This has been made possible by the header file iostream, where a No. of overloading def. for << are included.

```
for (int i = i; i < = 5; i + +)
for (int j = 1; j < = 5; j + +)
{
        cout << "The for matrix is << A [i] [j];
}
        for (int j = 1; j < = 5; j + +)
        for (int k = 1; k < = 5; k + +)
}
        cout << "The end matrix is << B [J] [K];
```

```
          int matrix multiplication (A, B)
          int C [k] [e] = A[i] [j] * B [J] [K];
          for (int k = 1; k < = 5; k + +)
          for (int C = 1; C < = 6; C ++)
     {
          cout << The result is C< C [K] [e];
     }.
```

(a) Operator + to concantinate two strings.

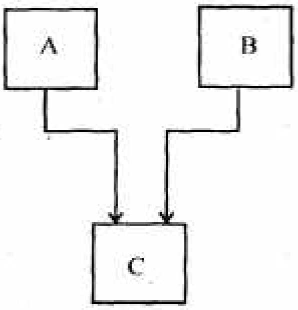
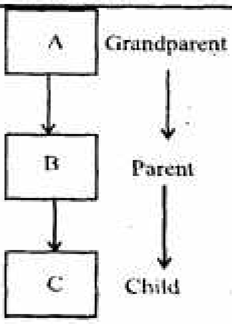
Q. 4. (b) Overload "*" operator to find the product of two matrices.

Ans.

```
     int add (int a, int b);
     int add (int a, int b, int c);
     double add (double x, double y);
     double add (int p) double q);
     double add (double p, int q);
     To overload * to find product of two matrix are as :
     # include <isotream.h>
     # include <conio.h>
     void main ( )
     void main ( )
     int matrix multiplication (int [i] [j], int B [j] [k]);
     int matrix multiplication (
};
```

Q. 5. (a) What do you understand by multiple and multilevel inheritance?

Ans. Multiple and Multilevel Inheritance :

Multiple	Multilevel
	

A derived class with only one base class and more than one base class and by only one derived class is called multiple inheritance.	The mechanism of during a class from another derived class is known as multilevel Interstance. Here first one is called grand parent then parent and after that we have child class.

Q. 5. (b) Differentiate between composition and inheritance.

Ans. Differentiate between composition & inheritance Inheritance : It means reusability. It is main features of OOP. It is always nice if we could refuse something that already exit rather than trying to create the same all over again. It would not only same time and money but also reduce frustration and increase reliability. For instance, the reuse of a class that has been already tested, debugged and used may times can same us the effort of developing and testing the same again.

Composition :

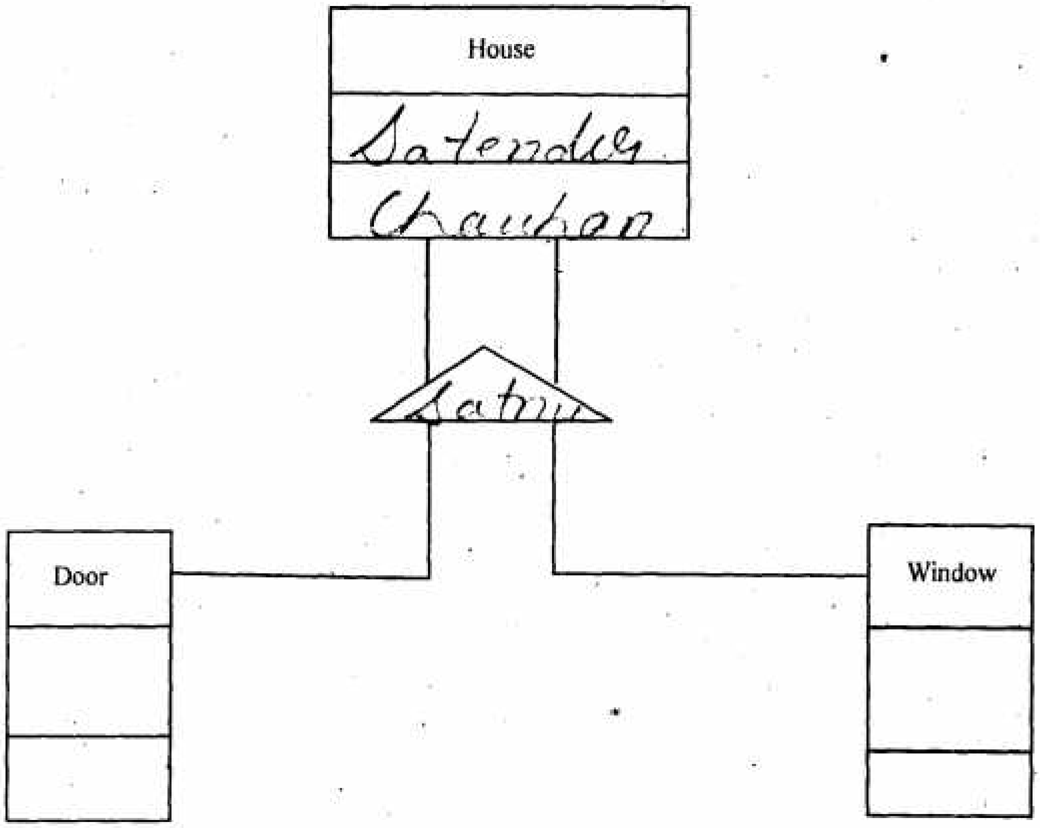


Fig. Composition Relationship

Q. 6. Write short notes on the following :

(a) Pure virtual function

(b) Abstract base class

(c) Dynamic binding.

Ans. (a) Pure virtual function :

It is normal practice to declare a function virtual inside the base class, and redefine it is the derived classes. The function inside the base class is seldom used for performing any task. It only saves as a place-holder. For eg.

Virtual void display ( ) = 0.

Such functions are called pure virtual functions.

(b) Abstract base class :

An abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class (to be in hinted by other classes). It is a design concept in program development and provide a base upon which other classes can be built.

(c) Dynamic binding :

It is one of powerful feature of C++. This require the use of pointers to objects. Dynamic binding is called late binding, it because the selection of appropriate functions is done dynamically at run time.

Q. 7. Give the description of following stream mode flags :

(a) std::ios::app

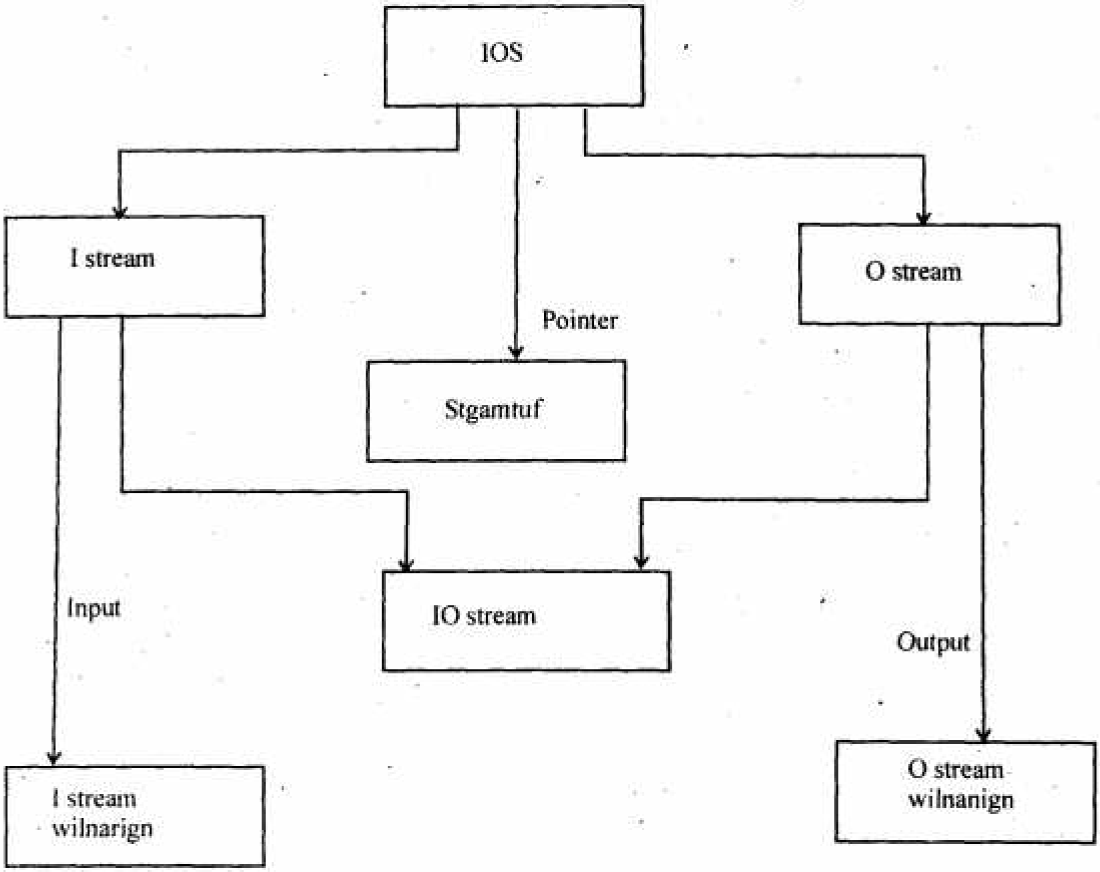
(b) std::ios::ate

(c) std::ios::binary

(d) std::ios::nocreate

(e) std::ios::noreplace.

Ans.



Ios : is declared as virtual base class, so that only one copy of its members are inherited by the isotream.

(a) ios:: app used for application

(b) ios ::ate : If used for general input output suppress class with data format.

(c) ios::binary : It inherits the properties of ios used to input binary data. It take input as get ( ) getline ( ), read ( ), we use extraction operator >>.

(d) std::ios::nocreate : The standard input output. Strain file is not created, it is used as an existing file.

(e) std::ios::noresplace : This is used as standard input output strength file, which is not replaced by another input/output files.

Q. 8. (a) Discuss the use of std::unexpected ( ) system function.

Ans. Standard file unexpected system function used for exception handling means for errors herding we used there st.file.

The two most common type of errors we used here logic here & syntactic errors. The logic errors occurs due to poor understanding of the problem and solution procedure. The synstactic errors gives due to poor understanding of lanuge itself. We can detect these errors by using debugging & testing procedures. These because are called exceptions. The use of standard unexpected used for error which are not cave under exceptions.

Q. 8. (b) What do you understand by Catch-all exception handlers? Explain.

Ans. Catch-all exception handless :

```
Catch ( ......)
{
        // staters exits for processing
        // all exceptions
}.
```

A catch statement to catch all exception instead of a certain tyre alone. This could be achieved by being the catch statement using above statement.

```
# include <iostream>
using namespace std;
{ void test (int x)
try {
if (x = = 0) throw X;
if (X = = -1) throw X;
if (X = = 1) throw 1.0;
}
        Catch ( ....)
        /// catch all
{
        Cout < < "Cought an exception.";
}
}.
```

Q. 8. (c) What is the use of class templates?: How can you supply default types for class template parameters? Explain.

Ans. Class Templates : Template is one of the features added to C++ recently. It is a new concept which enable as to define generic classes and functions and thus provides support for generic programming.

Generic program is an approach where generic types are used as parameters in algos. so that they work for a variety of suitable data types and data structures.

A template can be used to create a family of classes or functions. For eg. a class template for an array class.