	FINALTERM EXAMINATION FALL 2006 CS304 - OBJECT ORIENTED PROGRAMMING (Session - 1)	Marks: 50 Time: 120min

StudentID/LoginID: _____

Student Name: _____

Center Name/Code: _____

Exam Date: Friday, February 09, 2007

Please read the following instructions carefully before attempting any of the questions:

1. Attempt all questions. Marks are written adjacent to each question.

2. Do not ask any questions about the contents of this examination from anyone.

a. If you think that there is something wrong with any of the questions, attempt it to the best of your understanding.

b. If you believe that some essential piece of information is missing, make an appropriate assumption and use it to solve the problem.

c. Write all steps, missing steps may lead to deduction of marks.

d. All coding questions should be answered using the **C** ++ syntax.

e. Choose the most appropriate choice (among the given), while answering MCQs.

3. You are **not allowed to use IDE** (Dev-C++ or any) in your paper.

****WARNING: Please note that Virtual University takes serious note of unfair means. Anyone found involved in cheating will get an `F` grade in this course.**

For Teacher's use only											
Question	1	2	3	4	5	6	7	8	9		Total
Marks											

Question No: 1 (Marks: 2) - Please choose one

Which of the statements is true about destructor?

- Destructors are used to initialize data members
- Destructors return types are always void
- Destructors are called in reverse order of constructor called in a class hierarchy
- Destructors are used to display the data members of their respective class

Question No: 2 (Marks: 2) - Please choose one

Identify the type of variable that is part of a class, yet is not part of an object of that class.

- instance
- static
- global
- local

Question No: 3 (Marks: 2) - Please choose one

Which of the following operators can not be overloaded?

- ++
- new
- sizeof
- =

Question No: 4 (Marks: 2) - Please choose one

Target function for a call is selected at compile time in case of

- Static binding
- Polymorphism
- Encapsulation
- Dynamic binding

Question No: 5 (Marks: 2) - Please choose one

Suppose it is required to implement a single functionality with different data type, which of the following techniques should be used

- Inheritance
- Friend function
- Templates
- Encapsulation

Question No: 6 (Marks: 5)

Identify and correct the error(s) in the given code segment.

```
template< typename T >
class Parent {
        // …
};

template< >
class Child< int > : public Parent {
        // …
};

int main() {
        Parent< char > obj;
        Child< int > obj2;
        return 0;
```

```
}
```

Question No: 7 (Marks: 10)

Given are two classes A and B. class B is inherited from class A. Write a code snippet(for main function) that polymorphically call the method of class B. Also what changes do you suggest in the given code segment that are required to call the class B method polymorphically.

```
class A
{
public:
void method() { cout<<"A's method \n"; }

};

class B : public A
{

public:
void method() { cout<<"B's method\n"; }

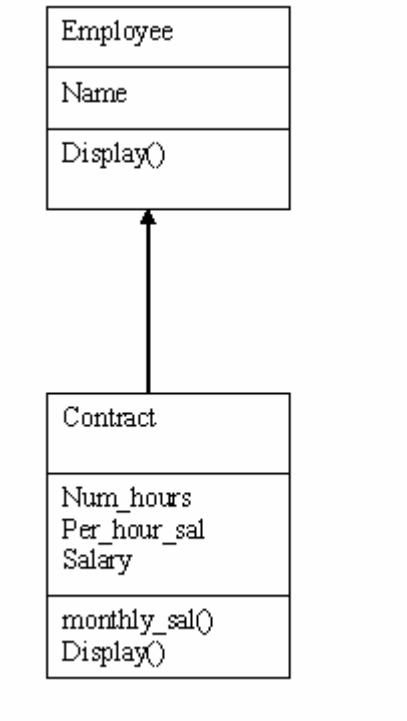
};
```

Question No: 8 (Marks: 10)

Implement the given class hierarchy where there exist a relation ship of inheritance between **Employee** and **Contract** (for contract employee). The **Display**() function of **Contract** should display the data that includes ; Name, Num_hours, Per_hour_sal, Salary

Monthly salary of contract employee is calculated as

Salary = Per_hour_sal*Num_hours



Question No: 9 (Marks: 15)

A Real world Problem:

This is a business rule of a bank that the Balance of customer must always be Positive,
Further that it is also a business rule that a customer can not withdraw amount greater than his/her current balance.

Question statement

You are required to write a C++ program to solve this real world problem, using **Exception Handling**.

Your program should consist of a class **Customer** and **4** instance variables. Cust_name, Cust_AC_No, Cust_balance, Amount_withdraw.

Cust_name : Customer Name
Cust_AC_No : Customer Account Number
Cust_balance : Customer Balance
Amount_withdraw : Amount that is to be withdraw

The class should further consists of getter and setter function for each instance variable.
The setter function of Cust_balance consists of a try and a catch block, if the user enters the balance less than zero (means negative, which is not possible) then this function should throw an exception, and its corresponding catch block should handle this exception by providing another opportunity to the user to enter the balance again.

Similarly the setter of Amount_withdraw variable should also have a try and a catch blocks. If the user enters the withdraw amount greater than the current balance then this function should throw an exception, and its corresponding catch block should handle this exception.

In main function just declare an object of **Customer** class and call all its setter and getter functions.