

CS304 Object Oriented Programming

Final Term Examination – Spring 2005

Time Allowed: 150 Minutes

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.

You are allowed to use the Dev-C++ compiler to write and test your code. If you do so please remember to copy and paste your code into the examination solution area. **(Do NOT share your code; your colleague could get higher marks than you!!)**

****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.**

Total Marks: 55

Total Questions: 03

Question No. 1

Marks : 10

```
class Exception {
```

```

protected:
    char message [30] ;
public:
    Exception() {strcpy(message,"Exception");}
    char * what() { return message; }
};

class DerivedException : public Exception {
public:
    DerivedException() {strcpy(message,"Derived exception");}
};

class A {
public:
    A(){ cout << "Constructor A\n";}
    ~A(){
        cout << "Destructor A\n";
        if(condition1) //condition one
            throw Exception();
    }
};

class B {
public:
    B(){ cout << "Constructor B\n";
        if(condition2) //condition two
            throw 1;
    }
    ~B(){ cout << "Destructor B\n";}
};

void Function1() {
    DerivedException DE;
    cout << "I am Function1" << endl;
    if(condition3) //condition three
    {
        throw DE;
    }
}

void Function2() {
    cout << "I am Function2\n" << endl;
    try{
        B obj2;
        Function1();
    }
    catch(int)
    {
        throw 10.1;
    }
    catch(...){

```

```

        throw;
    }
}

void Function3(){
    cout << "I am Function3\n";
    A obj;
    Function2();
}

int main() {
    try{
        Function3();
    }

    catch(DerivedException & e1){
        cout << "Exception: " << e1.what() << endl;
    }
    catch(Exception & e2){
        cout << "Exception: " << e2.what() << endl;
    }
    catch(int i){
        cout << "Exception: int";
    }
    catch(...){
        cout << "Exception: unknown\n";
    }

    cout << "End of Program\n";
    return 0;
}

```

- a.** What will be the output when only condition one is true? **02**
- b.** What will be the output when only condition two is true? **02**
- c.** What will be the output when only condition three is true? **02**
- d.** What will be the out put when all three conditions (1,2&3) are true? **02**
- e.** Make changes in the following code so that "Function" only throws the three types of exceptions mentioned in the body and no other exception. **02**

```

void Function(){
    if(condition1){
        ...
        throw Exception();
    }
    if(condition2){
        ...
        throw DerivedException();
    }
    if(condition3){

```

```

        throw 1;
    }
}

```

Question No. 2

Marks : 30

- a) The following program causes a compile-time error. Identify the error, explain the cause, and correct it.

10

```

template< typename T >
class Tree {
    // ...
};

template< >
class RBTre< int > : public Tree {
    // ...
};

int main() {
    Tree< char > chTree;
    RBTre< int > intRBTre;
    return 0;
}

```

- b) Consider the following vector class:

20

```

class Vector {
    ...
public:
    ...
    int* first();
    int* last();
    int* next( int* );
};

```

A reverse iterator is an iterator that traverses backwards for the operator ++, and forwards for the operator --. Write code for a reverse iterator that works with the following function:

(Note: typename T stands for iterator type and U for element type)

```

template< typename T, typename U >
T find( T start, T beyond, const U& x ) {
    while ( start != beyond && *start != x )
        ++start;
    return start;
}

```

Question No. 3**Marks : 15**

- | | |
|--|-----------|
| a) Briefly describe the two major issues in multiple inheritance? Illustrate by giving an example of each. | 04 |
| b) Describe the three essential characteristics of an <i>object</i> from the perspective of <i>object-oriented paradigm</i> . | 03 |
| c) Why use abstract classes at all? Why not just declare a class and then make sure you do not declare any variables of that type? | 02 |
| d) What is the difference between a partial and complete template specialization? Give an example of each. | 03 |
| e) What are the two mechanism of calling base class assignment operator from derived class assignment operator? | 03 |
-
-