

THE UNIVERSITY OF HONG KONG

FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

CSIS0396A Programming Methodology and Object-Oriented Programming
(for Software Engineering and Double Degree students)

Date: May 28, 2001

Time: 2:30 pm–5:30 pm

Candidates may use any calculator which fulfils the following criteria: (a) it should be self-contained, silent, battery-operated and pocket-sized; (b) it should have numeral-display facilities only and should be used only for the purpose of calculation; (c) it should not have any printing device, alphanumeric keyboard, or graphic display; and (d) it should not contain any recorded data or program. It is the candidate's responsibility to ensure that the calculator operates satisfactorily and the candidate must record the name and type of the calculator on the front page of the examination scripts. Lists of permitted/prohibited calculators will not be made available to candidates for reference, and the onus will be on the candidate to ensure that the calculator used will not be in violation of the criteria listed above.

Answer all questions in the answer book provided.

1. (Class interface design, 15%)

- a. Consider inheritance in the context of object-oriented programming. What is the relationship between a derived class and its base classes? What are the implications on the interface and data members of the two classes?

It is often said that in object-oriented programming, a square is *not* a rectangle. Suppose we have the following C++ class Rectangle:

```
class Rectangle {  
public:  
    Rectangle(double w, double h): width(w), height(h) {}  
    virtual void scale(double x, double y) { width *= x; height *= y; }  
    virtual double area() { return width * height; }  
private:  
    double width, height;  
};
```

- b. What are the problems to define a Square as a derived class of Rectangle? Is the interface of Rectangle appropriate for Square? Are the data members of Rectangle appropriate for Square?
- c. Suggest a better hierarchy for Square and Rectangle which allows their areas to be found in a uniform way. Draw the class diagram for your design, and write the C++ class interfaces for all classes in your design.

2. (Law of the Big Three, 15%)

- a. What is Law of the Big Three in C++?

Suppose we want to create a class RefCountedPointer to perform automatic deallocation of dynamic integer objects. The idea is as follow: whenever we want to keep a pointer, we

instead keep a `RefCountedPointer` type object. We make sure that the integer objects are deleted if and only if no `RefCountedPointer` refers to it.

A `RefCountedPointer` object contains a pointer to a structure `RefCountedRep`. The structure contains the real pointer and an integer called `count`. Whenever a `RefCountedPointer` is copied, the pointer to `RefCountedRep` is copied with the `count` incremented. Accordingly, whenever a `RefCountedPointer` is destroyed, the `count` is decremented, and the real pointer is deleted once the `count` becomes zero. Here are the definitions of the classes:

```
struct RefCountedRep {
    int count;
    int *real_ptr;
};

class RefCountedPointer {
    RefCountedRep *rep;    // private
public:
    RefCountedPointer(int *ptr = 0) {
        if (ptr == 0)
            rep = 0;
        else {
            rep = new RefCountedRep;
            rep->count = 1;
            rep->real_ptr = ptr;
        }
    }
    ~RefCountedPointer() {
        if (rep == 0) return;
        rep->count --;
        if (rep->count == 0)
            delete rep;
    }

    // Some more members needed to satisfy the law of the Big Three...

    int &GetContent() {
        if (!rep) throw 0;    // or something more meaningful
        return *(rep->real_ptr);
    }
};
```

- b. State the class invariants about the `RefCountedRep` objects.
 - c. Complete the `RefCountedPointer` class, following Law of the Big Three.
3. (*Object-oriented patterns, 15%*)

You are to design a file-manager type GUI application that allows users to view files and browse directories. You want to have a very flexible architecture, reading different things like ftp sites and tar-archives as directories. You also want to allow users to combine multiple such directories so that they are viewed as a single directory. Some classes are

already available for reading tar-archives and ftp sites.

State two object-oriented design patterns that are relevant to the application. For each of them, explain where you will use the pattern. Enumerate the classes in your design, and explain how they interact with the help of relevant class diagrams.

4. (*Separate compilation, 15%*)

As part of writing a Lisp interpreter with C++, a programmer writes the following header file `LispObject.h`, intended to be referred to by other source files with “`#include`” directives:

```
class LispObject {  
public:  
    virtual LispObject* eval();           // Evaluate the object  
    virtual void print() = 0;             // Print the object  
};  
  
class NilObject: public LispObject {  
public:  
    void print() { cout << "()"; }  
    LispObject* get_car();  
    LispObject* get_cdr();  
};  
  
LispObject* LispObject::eval() {  
    return this;  
}
```

```
NilObject nil_object;
```

The header file contains a number of problems, and the program cannot be compiled and linked. Find the problems, explain each of them, and show how to fix them.

5. (*CVS, 10%*) As of April 2001, the Mozilla (the open-source version of the Netscape browser) CVS tree looks like the following:

- a. Note that all the releases are in a CVS branch. Why did the Mozilla development team organize the CVS tree this way, rather than using a simple tag on the main trunk for each release?
- b. Suppose you are a Mozilla developer, working towards Mozilla 0.9 in the place marked as “we’re here” in the diagram above, with some modifications that are not ready for check-in yet. Now you suddenly need to fix a serious bug in Mozilla 0.8.1, with branch tag `rel-0-8-1`. Explain what you will do, state CVS commands as relevant. Include in your answer how to get back to the development towards Mozilla 0.9.

6. (*Error handling, 15%*)

Library writers usually need to communicate to the library user that an error has occurred.

- a. Apart from using language-supported exceptions, state one other strategy of error

handling, and compare their relative merits.

- b. The following function attempts to convert a link list of integers to an array so that a fast merge-sort can be applied. However, the merge-sort routine needs some memory, and if that memory is not available an exception is thrown. Explain why the function is not exception safe. Explain how you can make the function exception safe without using a try-catch block. (Hint: use the “resources acquisition is initialization” technique.)

```
struct int_list {
    int data;
    int_list *next;
};

void merge_sort(int[]);           // May throw

void sort_list(int_list *list, int size) {
    int *temp_arr = new int[size];
    int_list *curr = list;
    for (int i = 0; i < size; ++i) {
        temp_arr[i] = curr->data;
        curr = curr->next;
    }
    merge_sort(temp_arr);
    for (int i = 0; i < size; ++i) {
        curr->data = temp_arr[i];
        curr = curr->next;
    }
    delete [] temp_arr;
}
```

7. (Scripting, 15%)

This question is concerned with a scripting language called Python.

- a. Explain what is a Python extension module. What can be done by a Python extension module but not by a Python script? Is there any other reason to use a Python module?
- b. Other than using a Python module, how can C and C++ code be integrated with Python? Compare the strengths of the two methods.

END OF PAPER