Comprehensive Exam: Programming Languages Autumn 2005

This is a 60-minute closed-book exam and the point total for all questions is 60.

All of the intended answers may be written within the space provided. (*Do not use a separate blue book.*) Succinct answers that do not include irrelevant observations are preferred. You may use the back of the preceding page for scratch work. If you to use the back side of a page to write part of your answer, be sure to mark your answer clearly.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*

   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

By writing my "magic number" below, I certify that I acknowledge and accept the Honor Code.

_____

(Number)

Prob	# 1	# 2	# 3	# 4	Total
Score					
Max	20	12	12	16	60

10/31/05 REVISION

1. (*20 points*) ..... Short Answer

Answer each question in a few words or phrases.

(a) (***4 points***) In C++, it is possible to designate that a member function `f` is `virtual` in a base class, but not write this designation in the derived class. What relationship between a C++ base class `vtable` and derived class `vtable` implies that the function `f` is also virtual in the derived class? Explain.

(b) (***4 points***) Explain why this program, if compiled and executed without type checking, produces a run-time type error. At which point (and which line of code) will the Java compiler or run-time system report the error?

```
1: class A { ... amethod ...}
2:      class B extends A { ... bmethod ...}
3:      B[ ] bArray = new B[10]
4:      A[ ] aArray = bArray
5:      aArray[0] = new A()
6:      bArray[0].bmethod()
```

(c) (*4 points*) List three different reasons why Java programs generally run slower than C++ programs.

(d) (*4 points*) What is the main advantage of tail recursion optimization? Is it time or space saving? Explain.

(e) (*4 points*) The designers of C+++ are considering allowing functions to be declared inside any block. As a consequence, a function could be declared inside one function, and passed or returned to another function. What implementation costs are associated with this design option?

2. (*12 points*) ..... Scope and parameter passing and activation records

Consider this simple program:

```
1   int a :=  1
2   int b := 10
3   proc f(m) {
4      a := a+m
5      b := a+m
6   }
7   proc g(n) {
8      int a := 5
9      f(n)
10  }
11  g(a)
12  print a , b
```

(a) (*6 points*) Assume that all parameters are passed using call-by-value.

i. What values are printed on line 12 under static scoping?

ii. What values are printed under dynamic scoping?

(b) (*6 points*) Assume that only static scoping is used.

i. What values are printed if all arguments are passed using call-by-reference?

ii. What if call-by-value-result is used instead?

3. (*12 points*) ..... Templates and Generics

(a) (*4 points*) Consider a C++ class of this form:

```
template <class T>
class C<T> {
   ...
int f (T *x, T *y){ ...  x->less(y) ...}
   ...
};
```

Suppose a C++ program contains a an object of type `C<A>` for some C++ class `A`. Explain what functions, operators, and so on, class `A` must define in order for the code shown to compile and link correctly.

(b) (*4 points*) Suppose that an analogous class `C` is written in Java.

```
class C<T> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

Explain why the Java 1.5 compiler would not accept the code, based on the fragments shown.

(c) (*4 points*) The Java code can be written so that it will compile-time type check. Instead of `class C<T>`, we will need to write something of the form

`class C<T extends ...>`

What would you write in the underlined region to make this correct Java? Explain, and describe any additional classes or interfaces you may need.

```
class C<T extends  ____________________________> {
   ...
int f (T x, T y){ ...  x.compareTo(y) ...}
   ...
}
```

4. (*16 points*) ..... Concurrency

For the following question, we will use a very simple linked list class designed to be used by multiple threads. The class stores elements in a linked list and whenever an element is added or deleted, the code finds the maximum element in the list and stores it in a member variable. Assume that `max()` will never be called on an empty list.

```
public class CLinkedList
{
  protected CElement head_; //The list's head
  Comparable max_; //A reference to the maximum list element

  public CLinkedList()
  { //Constructor
    head_ = null;
    max_ = null;
  }

  //Assume max() will never be called on an empty list
  public synchronized Comparable max()
  {
    return max_;
  }

  //Add an element to the front of the list
  public synchronized void addElement(Comparable element)
  {
    CElement newHead = new CElement(element, head_);
    head_ = newHead;
    updateMax(); //Choose a new max element, if necessary
  }

  //Remove the first element from the list.
  public synchronized Comparable deleteFirst()
  {
    CElement deletedElement = head_;
    head_ = deletedElement.next; //Remove the first element.
    deletedElement.next = null; //Removed element points nowhere
    updateMax();                //Update the max element
    return deletedElement.element;
  }

  //Scans each element in the list until it finds an equal element or
  //the end of the list.  Returns true if the element is found.
  public boolean lookup(Comparable o){
    for (CElement i = head_; i != null; i = i.next){
      if (o.equals(i.element)){
        return true;
      }
    }
    return false;
  }

  //Scan every element in the list to find the maximum element.  Update
  //the max_ member to reference the maximum element.
  private void updateMax()
  {
```

```
  if (head_ != null){
    CElement tempMax = head_;
    for (CElement i = head_; i != null; i = i.next){
      if (tempMax.element.compareTo(i.element) < 0 ||
          tempMax.element.compareTo(i.element) == 0){
        tempMax = i;
      }
    }
    max_ = tempMax.element;
  }
}
//A simple list cell class.
private class CElement
{
  public CElement(Comparable element, CElement next){
    this.element = element;
    this.next = next;
  }
  public Comparable element; //The element referenced
                             //by this list element.
  public CElement next; //The next element in the linked list.
}
```

(a) (*5 points*) A race condition exists in the CLinkedList implementation because `lookup()` is not synchronized. In some cases, a call to `lookup()` may not find an object that is actually in the list. Give a sequence of calls to CLinkedList member functions by different threads that could result in `lookup()` failing to find an object that is in the list. Your answer should include a time-line of method calls made by specific threads and a description of what causes `lookup()` to fail.

(b) (*4 points*) Can a multi-threaded program become deadlocked when multiple threads concurrently operate on a single CLinkedList object? Explain.

(c) (*3 points*) If the programmer knows that CLinkedList will only be used on uniprocessor machines, is it necessary for `addElement()` and `deleteFirst()` to be synchronized?

(d) (***4 points***) Calling a synchronized method is more expensive than calling an unsynchronized method because a thread must acquire a lock before running in the monitor. If the programmer knows that `max()` will be called frequently, she would like to optimize the function by making it unsynchronized.

For this problem, keep in mind that the result returned by `max()` may be stale by the time the calling thread uses `max()`'s return value, which is correct. For example, thread 1 could call `max()` and get `max()`'s result, but before using the result thread 2 adds a new element to the list that becomes the new maximum value. The same line of reasoning applies to an unsynchronized version of `max()`.

i. Let's assume that the programmer knows that only `int` elements will be stored in CLinkedLists. Assuming that the programmer can rewrite any necessary code related to the type of elements stored in the linked list, can she make the `max()` method unsynchronized? Explain. For the purposes of this problem, assume that writes to integer variables are atomic, but writes to doubles are not atomic. If an operation is atomic, it either updates any necessary state and completes or it terminates without updating any state at all. If an operation is not atomic, it is possible that it can be interrupted in the middle of its execution and that some intermediate state might be visible to other parts of the program until it is resumed.

ii. If CLinkedList objects will hold only `double` values, can the programmer rewrite code and make `max()` unsynchronized? Explain.