CS 538 Final Exam

Thursday, May 19, 1994

2:45 PM – 4:45 PM CDT

1641 Humanities Bldg

Instructions

Answer *any four* questions. (If you answer more, only the first four will count.) Each question is worth 25 points. Please try to make your answers neat and coherent. Remember, if we can't read it, it's wrong. Partial credit will be given, so try to put something down for each question (a blank answer always gets 0 points!).

Question #1

(a) (15 points)
What is the role of the pseudo-variables self and super in SmallTalk? How do self and super differ?

Assume the following class and subclass definitions:

class GM
weight ↑ 3000
cost ↑ 4 ∗ self weight

class Saturn
superclass GM
weight ↑ 2000
cost ↑ 3 ∗ self weight
salePrice ↑ 0.9 ∗ super cost

Assuming myCar ← Saturn new, what does "myCar salePrice" return? Explain how you determined your answer.

(b) (10 points)
What must be done in C++ to get the effect of using self and super?

Question #2

Assume in C++ we have the following definition, which defines a class that sorts integer arrays (up to size 10000):

```
class sorter {
        char compare(int a,int b) { return a <= b;}; // Used by sort routine
        public:
            class array {
                public:
                  int data[10000];
                  int size;  number of elements of data that hold valid values
                  array () {size=0;};
            };

            // sort() is defined outside this class.
            // It sorts the in array into ascending order
            // using compare() and returns the sorted array
            array sort(array in);
        };
```

(a) (15 points) Assume we wish to create a derived class from sorter, revsorter, that sorts arrays into reverse (descending order). Show the changes needed in sorter and show your definition of revsorter. Note that you should *not* change sort() (because it may implement a tricky or complex sorting algorithm that's difficult to understand or modify).

(b) (10 points)
Our sort() routine currently passes and returns array objects which are fairly large. This may be a source of inefficiency. What can be done to make passing and returning array objects more efficient?

Question #3

(a) (12 points)
Define a SmallTalk class binNode that implements an ordered binary tree node. Your definition should contain instance variables representing the node's value, its left child and its right child. It should also contain an instance method inOrderDisplay that traverses the tree rooted by the node in inorder, printing out the value of each node. (Assume all objects that may be stored as the value of a binNode answer the unary message display by displaying their value in a suitable format.) (And yes, inorder means visit the left subtree, then the node, then the right subtree.)

(b) (13 points)
Extend your solution to (a) by adding a new instance method create: aList. aList is a sorted list. create: should create a balanced binary tree (composed of binNodes) corresponding to the elements of aList. A balanced binary tree has the property that each subtree in the tree contains the same number of nodes (± 1) as its sibling. You may assume that SmallTalk lists implement the usual set of list operations (length, head, tail, cons, reverse, append, etc.).

Question #4

Explain how type-inference works in ML. Illustrate the process by solving for the type of the following function:

```
fun pm _ [] = []
 |  pm f ((a,b)::t) = (f(a,b))::pm f t
```

What does this function do?

Question #5

(a) (13 points)

Assume we wish to create a datatype (or object) called pair that contains a pair of values. pairs should implement three operations:

- A constructor (call it what you like) that takes a pair of values and returns a pair.
- first, which takes a pair and returns its first value.
- second, which takes a pair and returns its second value.

Assume that pair is *monomorphic*—that is, the types of values in all pairs are always the same.

Show how to define pairs in any two of the following languages: SML, SmallTalk, C++.

(b) (12 points)

Now assume that pairs may be *polymorphic*—that is, in the same program pairs may sometimes hold one pair of types and sometimes hold a different pair of types.

Show how your solutions to part (a) must be changed to support polymorphic pairs.

Question #6

(a) (15 points)

In C arrays are very simple. In fact, they are often *too* simple. In particular, since they have static bounds, it is often difficult to choose an appropriate size.

Flex arrays, which grow in size as needed, are easy to program in C++. Initially a flex array has zero elements. If a subscript beyond the array's current size is encountered, the flex array simply grows to allow that subscript. A subscript is never too large; the flex array expands when necessary (for simplicity, negative subscripts are still invalid). An outline of a C++ class, flexarray, is shown below. It has a constructor and destructor, as do most classes. Function size() returns the array's current size. If size() = s, then space for s elements has been allocated in the array. operator[] is the subscripting operator. When given a non-negative subscript, i, a reference to the i-th element of the flex array is returned. If i is too big, the flex array expands itself.

Complete the definition of flexarray by filling in the private data definitions needed, and the bodies of the four functions.

```
template<class T>
class flexarray{

    public:
     flexarray(){};
     ~flexarray(){};
     int size() {};
     T&  operator[](const int i){};
};
```

(b) (10 points)

Though C does not allow arrays to be assigned, there is no reason to disallow assignment of flexarrays. If you do not define assignment for flexarrays, the fields of the class will simply be copied, just as in the assignment of structs. Is this satisfactory for your implementation of flexarrays? If it is explain why. If it isn't, complete the following definition of the assignment operator, which would be included in the definition of flexarray's class:

```
void  operator=(flexarray<T> rhs){};
// lhs of assignment is the object this function sits in
```

Question #7

(a) (12 points)
Let F and G be ML functions and let L be a list containing values $[v_1, v_2, \ldots, v_n]$. Write an ML function munge(F,G,L) that computes a list with the following $2 \times n$ values: [(F v_1), (G v_n), (F v_2), (G v_{n-1}), $\cdots$ (F v_n), (G v_1)]

(b) (13 points)
Let F and G be ML functions and let L be a list containing values $[v_1, v_2, \ldots, v_n]$, where n is even. Write an ML function mangle(F,G,L) that computes a list with the following n values:
[(F v_1), (G v_2), (F (F v_3)), (G (G v_4)), (F (F (F v_5))), (G (G (G v_6))), $\cdots$]