Computer Science & Engineering 505 – Midterm

November 10, 1993

Open papers and notes – 50 minutes – 120 points total

Name: _____

Please write any long answers on the back of the exam or on a separate piece of paper.

1. (10 points) Consider the following program in an Algol-like language.

```
begin
integer n;

procedure pine(n: integer);
  begin
  oak;
  end;

procedure oak;
  begin
  print("in procedure oak");
  print(n);
  end;

n := 10;
pine(1);
print("in main program");
print(n);
end;
```

   (a) What is the output if lexical scoping were used (as would be the case in an Algol-like language)?

   (b) What is the output if dynamic scoping were used?

2. (15 points) Consider the following program in an Algol-like language.

```
begin
integer barnacle;
procedure clam(j,k: integer);
    /* note that the formal parameters may be aliased */
    begin
    barnacle := barnacle+10;
    j := j+1;
    k := k+2;
    print(j,k,barnacle);
    end;
barnacle := 1;
clam(barnacle, barnacle);
print(barnacle);
end;
```

   What is the output if `j` and `k` are both passed by:

(a) call by value

(b) call by value-result

(c) call by reference

If the results may be implementation-dependent, explain why, and explain the reasoning behind your answer.

3. (30 points) Suppose that the following Miranda script has been filed in. (Recall that \/ is the "or" operator.)

```
tree * ::= EmptyTree | Node * (tree *) (tree *)

sample = Node 1 (Node 10 EmptyTree EmptyTree) EmptyTree

compose f g x = f (g x)

my_map f [] = []
my_map f (a:x) = f a : my_map f x

double x = 2*x

my_foldr op r = f
                where
                f [] = r
                f (a:x) = op a(f x)


my_sum = my_foldr (+) 0

tree_map f EmptyTree = EmptyTree
tree_map f (Node x left right) = Node (f x) (tree_map f left) (tree_map f right)

tree_fold op r = f
                 where
                 f EmptyTree = r
                 f (Node x left right) = op x (op (f left) (f right))



tree_sum = tree_fold (+) 0

tree_or = tree_fold (\/) False

tree_member tree x = tree_or (tree_map (= x) tree)

many_numbers n = Node n (many_numbers (n+1)) (many_numbers (n+1))
```

What is the result of evaluating the following Miranda expressions? If there is a compile-time type error, or a run-time error, or a non-terminating computation, say so. If the result is an infinite structure, give some initial parts of it. If the expression is followed by `::`, then give the type, instead of the value.

(a) `my_map double ::`

(b) `compose ::`

(c) `compose double ::`

(d) `tree_fold ::`

(e) `tree_sum sample::`

(f) `tree_sum (many_numbers 1)::`

(g) `many_numbers 10`

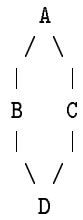
(h) `tree_map (= 11) (many_numbers 10)`

(i) `tree_member (many_numbers 10) 11`

(j) `tree_member (many_numbers 10) 5`

4. (10 points) Suppose we have a prototype-based object-oriented language, with an object `p` that is a child of a prototype `point`. If `p` gets a message it doesn't understand, it delegates it to its parent. Why do we need to use delegation? If `p` gets a message such as `move` that it doesn't understand, why can't it simply send the `move` message to `point`, without using delegation?

5. (10 points) Consider the following class hierarchy in a system with multiple (implementation) inheritance, so that `B` and `C` are subclasses of `A`, and `D` is a subclass of `B` and `C`.



Suppose that a method `clam` is defined for class `A`, and that the method `squid` is defined for classes `B` and `C`.

(a) What would happen if you send the message `clam` to an instance of `D` in Extended Smalltalk? What would happen if you send the message `squid` to an instance of `D`? Explain why.

(b) Similarly, but in CLOS?

(c) Similarly, the tree-oriented system Snyder proposes?

In each case explain why.

6. (15 points) Suppose we have an object-oriented language with an abstract type system using the contravariant rule. Suppose we have an abstract type `ReadStream`, parameterized by `T`, the type of object in the stream. `ReadStream` has two operations: `next` and `atEnd`. `next` takes no arguments and returns the next item in the stream. `atEnd` takes no arguments and returns a boolean. There is also a type `WriteStream`, with three operations: `nextPut`, `nextPutAll`, and `atEnd`. `nextPut` takes one argument (the item to append to the stream) with no return value. `nextPutAll` takes one argument (an array of items, each of which is to be appended to the stream), and returns no value. `atEnd` takes no arguments and returns a boolean.

Suppose that type `Character` is a subtype of `Object`, and `Integer` is a subtype of `Number`, which is a subtype of `Object`.

(a) What is the type of the `nextPutAll` operation?

(b) What is the relation between `WriteStream[Object]` and `WriteStream[Character]`?

(c) What is the relation between `ReadStream[Object]` and `ReadStream[Character]`?

(d) Suppose we have a new type `ReadWriteStream` that implements the `next`, `atEnd`, `nextPut`, and `nextPutAll` operations. What is the relation among `ReadStream[Character]`, `WriteStream[Character]`, and `ReadWriteStream[Character]`?

7. (10 points) Compare the issues that arise in multiple (abstract) subtyping as compared with multiple (implementation) inheritance.

8. (10 points) Compare single dispatching with multiple dispatching in object oriented languages. What are the advantages and disadvantages of each?

9. (10 points) True or false?

(a) An advantage of Turner's SK machine is that the code is naturally self-optimizing. For example, given a call to `factorial 5`, the system will automatically remember this result and use it the next time `factorial 5` is evaluated.

(b) In Smalltalk-80, when an ordinary object is sent a message, the system first looks in the class of that object for a corresponding method. When a class is sent a message, such as `new`, the system first looks in its metaclass for a corresponding method.

(c) One of the criticisms that Snyder makes of graph-oriented inheritance systems is that they make a class's superclasses part of the interface it presents to its subclasses.

(d) In Algol-60, procedures can be passed as parameters, but a procedure name can't be assigned to a global variable. This restriction allows Algol to use stack-based storage.

(e) Last year's midterm included the following question:

> True/false questions have no place on an examination in a graduate course; the only time they are given is when instructors are too lazy to grade penetrating page-long answers to essay questions.