

CS 345 - Programming Languages
Spring 2008

FINAL

May 13, 2008

DO NOT OPEN UNTIL INSTRUCTED

YOUR NAME: _____

Collaboration policy

No **collaboration** is permitted on this exam. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Sciences department code of conduct can be found at <http://www.cs.utexas.edu/users/ear/CodeOfConduct.html>

Final (125 points)

Problem 1 (12 points)

Define the following terms:

Stack smashing:

Pass-by-reference:

Dynamic typing:

Parametric polymorphism:

Problem 2

Problem 2a (5 points)

What is the purpose of *lexical analysis*? What type of abstract machine is used to implement it?

Problem 2b (5 points)

What is the purpose of *syntax analysis*? What type of abstract machine is used to implement it?

Problem 3

Problem 3a (3 points)

In bottom-up parsing, what is the primary cause of a shift-reduce error?

Problem 3b (3 points)

Give an example of a syntactic statement used in modern programming that can cause this parsing error to occur? How is it resolved?

Problem 4 (7 points)

Rewrite the following ambiguous BNF grammar for an arithmetic expression language so that it is unambiguous with respect to the standard operator precedence and associativity rules.

$$E ::= E + E \mid E - E \mid E * E \mid E / E \mid (E)$$

Problem 5 (8 points)

List **three** garbage collection techniques. Which technique results in the worst memory paging behavior? Which technique results in the least memory fragmentation over time?

Problem 6

Consider the imperative and the functional programming styles.

Problem 6a (3 points)

What is the main advantage of programming *without* side effects?

Problem 6b (4 points)

What are **two** advantages of programming *with* side effects?

Problem 7

Consider the following ML function that constructs a new association list by calling a function on each of the values in the association list (in this function, `hd` returns the head of a list, `tl` returns the tail):

```

fun key(k,v) = k;
fun value(k,v) = v;
fun mapValues f alist =
  if null alist then nil
  else (key(hd alist),f(value (hd alist))) :: mapValues f (tl alist);

```

Here are the types inferred for `key` and `value`, respectively:

```

key: ('a * 'b) -> 'a
value: ('a * 'b) -> 'b

```

Here is how `mapValues` might be used:

```

mapValues double [("bob",4), ("betty",7), ("jane",6)] =>
  [("bob",8), ("betty",14), ("jane",12)]

```

Problem 7a (7 points)

Explain how the ML type inference algorithm would compute the type of the `mapValues` function.

Problem 7b (4 points)

What is the type inferred for the `mapValues` function?

Problem 8

Consider the following recursively defined Scheme function, where `list2` is a function that returns a 2-element list:

```

(set zip (lambda (l1 l2)
  (if (or (null? l1) (null? l2)) nil
      (cons (list2 (car l1) (car l2))
            (zip (cdr l1) (cdr l2))))))

```

The `zip` function takes two lists and returns a list of 2-element lists. For example,
`(zip '(3 4 5) '(hi there sue sam)) => '((3 hi) (4 there) (5 sue))`

Problem 8a (7 points)

Write `zip` in ML using pattern matching. The result should be a list of 2-element tuples. You may assume that the input lists are of equal length. Use the following implementation of `length` as a guide:

```
fun length [] = 0
  | length (x::xs) = 1 + length xs;
```

Problem 8b (4 points)

The type of `length` is `'a list -> int`. What is the type of `zip`?

Problem 9

Consider implementing a polynomial of a single variable as a list of coefficients, in increasing order of degree. For example, the polynomial $2x^3 + 5x - 4$ would be represented using the list `(-4 5 0 2)`.

Here is a Scheme function that takes a polynomial and an integer constant and returns a new polynomial that has each coefficient multiplied by the constant:

```
(define poly* (p i)
  (if (null? p) nil
      (cons (* (car p) i)
            (poly* (cdr p) i))))
```

Problem 9a (2 points)

Is the above implementation of `poly*` tail-recursive? Explain.

Problem 9b (2 points)

What is the main advantage of implementing `poly*` so that it is tail-recursive?

Problem 9c (6 points)

Rewrite the implementation of `poly*` using a standard higher-order function that we discussed in class and an anonymous `lambda` expression.

Problem 9d (3 points)

What are *free* variables?

Problem 9e (3 points)

What are the free variables of the anonymous `lambda` expression in your solution to Problem 9c?

Problem 9f (3 points)

What is *lexical scoping*?

Problem 9g (3 points)

Why is lexical scoping crucial in your solution to Problem 9c?

Problem 10

Problem 10a (3 points)

Consider the following Prolog implementation of `append`:

```
append([X|Xs], Ys, [Z|Zs]) :- append(Xs,Ys,Zs), X=Z.  
append([], Ys, Ys).
```

Why is this implementation potentially problematic?

Problem 10b (5 points)

Consider the following implementation of addition to a set, represented as a list with no duplicates.

```
add([], X, [X]).  
add([X|Xs], X, [X|Xs]).  
add([X|Xs], Y, [X|Zs]) :- not(X=Y), add(Xs,Y,Zs).
```

What is the effect of the following query? Explain.

```
?- add([4,5],X,[3,4,5]).
```

Problem 11

Problem 11a (3 points)

If a type A is known to be a subtype of a type B , what does this enable programs to do that they could not do in a non-object-oriented language?

Problem 11b (3 points)

What is the difference between message passing and overloading?

Problem 11c (3 points)

What is the difference between a virtual and a non-virtual function in C++?

Problem 12

Consider the following C++ code:

```
class Point {
public:
    int x;
    int y;
    Point(int x1, int y1) { x = x1; y = y1; }
    Point* move(int dx, int dy) {
        return new Point(x+dx,y+dy); }
};
class Point3D: public Point {
public:
    int z;
    Point3D(int x1, int y1, int z1): Point(x1,y1) { z=z1; }
    Point3D* move_3D(int dx, int dy, int dz) {
        return new Point3D(x+dx,y+dy,z+dz); }
};
```

```
Point *p = new Point(3,4);
Point3D *p1 = new Point3D(3,4,5);
```

Problem 12a (10 points)

For each of the following statements, indicate whether it is statically type-correct (under the type checking rules of C++) and, if not, explain why not. Each statement should be considered to follow the previous statements.

```
Point *p2 = new Point3D(3,4,5);
```

```
Point3D* p3 = p2;
```

```
Point3D* p4 = p2->move_3D(7,6,5);
```

```
Point* p5 = p1->move_3D(-1,-2,4);
```

```
Point* p6 = p1->move(1,-1);
```

Problem 12b (4 points)

Consider adding the following method to the `Point3D` class:

```
class Point3D: public Point {
    ...
    void move(int dx,int dy) {
        x+=dx; y+=dy; }
}
```

How would the C++ static type checker react to this new method? Why? If there are any problems, how could we change this code to fix them?