Announcements

- HW12: Transaction server
  - Due today at 2pm
  - You can use up to 2 late days, as always

Final Exam is Cumulative

- Programming Language Syntax (Ch. 2.1-2.3.3)
- Logic Programming and Prolog (Ch. 12)
- Scoping (Ch. 3.1-3.3)
- Programming Language Semantics (Sc. Ch. 4.1-4.3)
- Functional programming (Ch. 11)
  - Scheme and Haskell, map/fold questions
  - Lambda calculus (Ch. 11 Companion)
  - Data abstraction: Types (Ch. 7, 8)
  - Control abstraction: Parameter Passing (Ch. 9.1-9.3)
  - Object-oriented languages (10.1-10.2)
  - Scala
  - Concurrency (13.1), “What can go wrong?” questions
  - Dynamic languages (Ch. 14 optional)
  - Comparative Programming Languages

Practice Problems (Quiz 7)

type Name = String

data Expr = Var Name
| Val Bool
| And Expr Expr
| Or Expr Expr
| Not Expr
| Let Name Expr Expr

Fill in the type signature of find:

- Looks up variable n in binding environment env.
- Returns first binding or throws Exception if no binding of n.
- Ex: find “x” [(“x”, True), (“y”, False), (“z”, True)] returns True

find :: _________________________________

find n env = head [ bool | (var, bool) <- env , var == n ]
Practice Problems (Quiz 7)

Fill in the Or and Let arms of eval:
-- Purpose: evaluates expression e in binding environment env
-- Returns the boolean value of e or throws an Exception
-- Ex.: eval (Var "x") [("x",True),("x",False)] returns True

eval :: Expr -> [(Name,Bool)] -> Bool

val e env |

case e of
  Var n -> find n env
  Val b -> b
  And e1 e2 -> (eval e1 env) && (eval e2 env)
  Or e1 e2 -> (eval e1 env) || (eval e2 env)
  Not e1 -> not (eval e1 env)
  Let n e1 e2 -> eval e2 ((n,(eval e1 env)):env)

Practice Problems (Quiz 8)

In programming languages, types and type checking

(a) Prevent runtime errors
(b) Abstract data organization and implementation
(c) Document variables and subroutines
(d) All of the above

Quiz 8 Question 2

Let A denote all syntactically valid programs.
Let S denote all syntactically valid programs that execute without forbidden errors. Let T denote all programs accepted by certain type-safe static type system. Which one best describes the relation between T, S and A?

(a) $T \subseteq S \subseteq A$
(b) $T \subseteq S \subseteq A$
(c) $T \subseteq S \subseteq A$
(d) $T \subseteq S \subseteq A$

Practice Problems (Quiz 8)

Again, let S denote all syntactically valid programs that execute without forbidden errors. Let $T'$ denote all programs accepted by certain type-unsafe static type system.

$T' \nsubseteq S$ is
(a) true
(b) false
Practice Problems (Quiz 8)

**int w[10]()** is an invalid declaration in C. Why?

(a) true  
(b) false

In C, functions are third-class values. Thus, we cannot pass a function as argument, return a function as a result, or assign a function value to a variable, or structure.

Practice Problems (Quiz 8)

**w** in declaration **int (*w[10])()** is

(a) A function  
(b) An array  
(c) A pointer

Practice Problems (Quiz 8)

**A** is a 3-dimensional array of **ints**:  
**int [0..1,0..1,0..1] A.**  
The elements are ordered in memory as:  
A[0,0,0], A[1,0,0], A[0,1,0], A[1,1,0], A[0,0,1], A[1,0,1], A[0,1,1], A[1,1,1]  
This is

(a) Column-major order  
(b) Row-major order  
(c) Neither

Practice Problems (Quiz 8)

**typedef struct { int *i; char c; } huge_record;**

**void const_is_shallow(  
    const huge_record* const r) {**
    **int *x = r->i; // or just *(r->i) = 0;**
    **x = 0;**
    **}**

Is this a compile time error?  
No.

Practice Problems (Quiz 9)

```java
class Account {
    int ammount = 0;
    void deposit(int x) {
        ammount = ammount + x;
    }
}

class DepositTask implements Runnable {
    public void run() {
        synchronized (this) {
            Main.act.deposit(10);
        }
    }
}

class Main {
    static Account act = new Account();
    public static void main(String arg[]) {
        ExecutorService pool = Executors.newCachedThreadPool();
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.shutdown();
    }
}
```

Question 1.  
What are all possible values for **act.ammount** at the ends?  

Question 2.  
Instance field **ammount** in **Account** has

(a) private visibility  
(b) protected visibility  
(c) public visibility  
(d) package visibility

YOUR RCSID:  
AND NAME:

Programming Languages Quiz 9  
Monday May 2, 2016  
8 points total

Questions 1 and 2 refer to the Java code below:

```java
class Account {
    int ammount = 0;
    void deposit(int x) {
        ammount = ammount + x;
    }
}

class DepositTask implements Runnable {
    public void run() {
        synchronized (this) {
            Main.act.deposit(10);
        }
    }
}

class Main {
    static Account act = new Account();
    public static void main(String arg[]) {
        ExecutorService pool = Executors.newCachedThreadPool();
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.shutdown();
    }
}
```
Practice Problems (Quiz 9)

public class Main {
    static Account act = new Account();
    public static void main(String arg[]) {
        ExecutorService pool = Executors.newCachedThreadPool();
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.execute(new DepositTask());
        pool.shutdown();
        pool.awaitTermination(60, TimeUnit.SECONDS);
    }
}

What can act.ammount be at the end? 10,20,30

Practice Problems (Quiz 9)

Generic Java class MyList is defined as:

public class MyList<T extends Number> {
    // Type parameter T is bounded
    // MyList uses interface of Number,
    // e.g., intValue(), doubleValue()...
}

Is MyList<Integer> valid? Yes.
Is MyList<String> valid? No.

Practice Problems (Quiz 9)

Regular expressions
Context-free grammars
- Derivation, parsing, ambiguity, precedence
- LL(1) grammars and parsing
  - FIRST, FOLLOW, PREDICT sets, LL(1) parsing table
  - Obstacles to LL(1)-ness
- SLR(1) grammars and parsing
  - CFM
  - Conflicts in SLR(1)

PL Syntax Problems

Problem 1. Given grammar
Start → S
S → TS | [ S ] S | ε
T → ( X )
X → TX | [ X ] X | ε

Fill in the FIRST and FOLLOW table:
FIRST
S
T
X
FOLLOW

Is this grammar an LL(1) grammar?

Programming Language (PL) Syntax

PL Syntax Problems

Problem 2. Augmented grammar
S → E
E → E + E
E → y

Construct the CFM and add the “reduce by” labels. Is the grammar a SLR(1) grammar?

Prolog, Scoping

Prolog --- expect simple programming questions, as in practice tests and Exam 1
Scoping
- Static scoping
- Dynamic scoping with deep and shallow binding
Scoping when functions are first-class values
  - Scoping questions on code in Scala and Python
  - Exam 2
  - Jensen’s device in Scala, etc.
### Programming Language Semantics

- Attribute grammars
  - Attributes
  - Synthesized vs. inherited
- Semantic rules
- S-attribute grammars
- L-attribute grammars
  - Only terms, no problems that involve L-attribute grammars

### Attribute Grammar Problems

- Problem 1
  Given CFG
  \[
  E \rightarrow E + T | T \\
  T \rightarrow T * F | F \\
  F \rightarrow (E) | id
  \]
  Write an attribute grammar which computes at the root of the tree a `count` containing the maximum depth to which parentheses are nested in the expression

### Scheme

- Recursion
  - Tail recursion
  - Shallow and deep recursion
- Higher-order functions
  - `map/fold` problems
- Scoping in Scheme

### Scheme Problems

- Problem 1
  Consider the following function:
  
  ```scheme
  (define fun (lambda (n)
    (if (= n 1) 0 (+ 1 (fun (quotient (+ n 1) 2))))))
  ```
  What does this function do?
  Is it tail-recursive? If not, write a tail recursive version of this function.

- Problem 2
  Define a function `filter` which takes a list and returns a list containing all elements that satisfy a given predicate. For example,
  
  ```scheme
  (filter (lambda (x) (< x 5)) '(3 9))
  ```
  should return `(3)`. The entire body of `filter` should be one call to `foldl`.
  
  ```scheme
  (define (filter p lis)
    (foldl __________________________ lis))
  ```

### Scheme Problems

- Problem 2
  Define a function `filter` which takes a list and returns a list containing all elements that satisfy a given predicate. For example,
  
  ```scheme
  (filter (lambda (x) (< x 5)) '(3 9))
  ```
  should return `(3)`. The entire body of `filter` should be one call to `foldl`.
  
  ```scheme
  (define (filter p lis)
    (foldl __________________________ lis))
  ```
More Scheme Problems

- Other problems
  - Problems from sample final
  - Part 1 from practice problem set

Haskell Problems

- Write all Scheme problems in Haskell too!

Lambda Calculus

- Syntax and semantics
- Free and bound variables
- Rules of lambda calculus
  - \( \alpha \)-conversion
  - \( \beta \)-reduction
- Evaluation order
  - Applicative order reduction
  - Normal order reduction
- Normal forms: WHNF, HNF, NF

Lambda Calculus Problems

- Problem 1 (from sample final). Reduce the following term until no more reductions are possible:
  \[ (\lambda x. x x) ((\lambda y. y) (\lambda z. z)) \]
- Problem 2 (from sample final). Suppose we defined the following combinators:
  \[ T = \lambda x. \lambda y. x \quad \text{"T stands for true"} \]
  \[ F = \lambda x. \lambda y. y \quad \text{"F stands for false"} \]
  \[ N = \lambda x. (x \ F \ T) \quad \text{"N stands for not"} \]

  Show that \( N \ (N \ z) \rightarrow_{\beta} z \) where \( z \) is either \( T \) or \( F \)

Rules of Lambda Calculus: Exercises

- Use \( \alpha \)-conversion and/or \( \beta \)-reduction:
  \[ (\lambda x. x) y \rightarrow_{\alpha\beta} ? \]
  \[ (\lambda x. x) (\lambda y. y) \rightarrow_{\alpha\beta} ? \]
  \[ (\lambda x. \lambda y. \lambda z. x \ z \ (y \ z)) (\lambda u. u) (\lambda v. v) \rightarrow_{\alpha\beta} \]

  Notation: \( \rightarrow_{\alpha\beta} \) denotes that expression on the left reduces to the expression on the right, through a sequence \( \alpha \)-conversions and \( \beta \)-reductions.

Normal Forms

- \( \lambda z. z \ z \) is in NF, HNF, or WHNF?
- \( (\lambda z. z) (\lambda x. x) \) is in?
- \( \lambda x. \lambda y. \lambda z. x \ z \ (y \ (\lambda u. u)) \) is in?
- \( (\lambda x. \lambda y. x) z ((\lambda x. z \ x) (\lambda x. z \ x)) \) is in?
- \( z ((\lambda x. z \ x) (\lambda x. z \ x)) \) is in?
- \( \lambda z. (\lambda x. \lambda y. x) z ((\lambda x. z \ x) (\lambda x. z \ x)) \) is in?
Lambda Calculus Problems

- Other problems
  - Go over problems from HW8

Definitions

- What is a closure, higher-order function, tail recursion?
- What is normal-order reduction, applicative-order reduction?
- What does “functions are first-class values” mean?
- Weak-head normal form (WHNF), head normal form (HNF), normal form
- Other...

Types and Type Equivalence

- Type construction
- Type equivalence
  - Structural equivalence
  - Name equivalence
    - Loose name equivalence, strict name equivalence
  - Type equivalence in C

Types Problems

- Problem 1 (from HW10). Consider the following type declaration in C:
  ```
  double (*foo(double (*)(double, double[]), double))()
  ```
  Using appropriate type constructors (pointerTo, array and →), show the type tree for foo

Other Problems

- Go over problems from HW8

Types Problems

- Other sample problems
  - Go over HW10
  - Problem 6 from sample final
  - Problems from practice problem set

Definitions

- What is a type system?
- What is type safety?
- What is statically typed language?
- Dynamically typed language?
- What is recursive type?
- What is structural type equivalence?
- What is name type equivalence
  - Loose name equivalence, strict name equivalence
Parameter Passing Mechanisms

- Call by value
- Call by reference
- Call by value-result
- Call by name

Parameter Problems

- Problem 1 (from practice problem set).
  Write a single program fragment that produces different result under each of the following parameter passing mechanisms
  - Call by value
  - Call by reference
  - Call by value-result
  - Call by name

Parameter Passing Problems

```plaintext
x : int
b : array [0..1] of int

procedure p(y,z : int)
begin
  y := y+1
  print x,z
end

x := 0;
b[0] := 0; b[1] := 1;
p(x,b[x]);
print x;
```

Parameter Passing Problems

- Other problems
  - Problem 5 from sample final
  - Problems from practice problem set

Note: All questions on parameter passing assume value model for variables, unless explicitly specified otherwise

Definitions

- What is the value model for variables?
  - Give examples of languages that use the value model for variables

- What is the reference model for variables?
  - Give examples of languages that use the reference model for variables

Object-oriented Programming

- Benefits of object-oriented programming
  - What is subtype polymorphism?
    - What is dynamic binding?

- What is parametric polymorphism?
  - Explicit parametric polymorphism, implicit parametric polymorphism?
Concurrency

- Definitions
  - Message passing vs. shared memory
  - Shared mutable state
  - Atomic action
  - Data race, atomicity violation, deadlock
  - Memory consistency models
  - Actor model

- “What can go wrong” questions

Concurrency Problems

- Other problems
  - Part 6 from practice problem set
  - Examples we did in class

Other Topics

- Comparative programming languages
  - Problem 7

THANK YOU!

- We started at about 330 we are now at 311

- Thanks to Barb Cutler and all of you who have contributed to Submitty

- Thanks to Sam, Mike, Varun, Bill, Jordan and Kasra