Announcements

- HW9: Java concurrency
  - Due Wednesday 6PM
  - Tests available in the HW Server
  - Include a README file with your full names

- Questions?

Final Exam is Cumulative

- Programming Language Syntax (Ch. 2.1-2.3.3)
- Logic Programming and Prolog (Ch. 11)
- Scoping (Ch. 3.1-3.3)
- Programming Language Semantics: Attribute Grammars (Sc. Ch. 4.1-4.3)
- Functional programming and Scheme (Ch. 10)
  - map/fold problems as in Exam 2
- Lambda calculus (notes + Ch. 10.6 on CD)
- Data abstraction: Types (Ch. 7.1 – 7.4)
- Control abstraction: Parameter Passing (Ch. 8.1-8.3)
- Object-oriented languages (9.1-9.2)
- Concurrency (12.1-12.3) What can go wrong questions
- Dynamic languages (Ch. 13 optional)
- Comparative Programming Languages

Practice Problems (Quiz 7)

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
Practice Problems (Quiz 7)

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

Quiz 7 Question 2

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.

$T' \not\subseteq S$ is

(a) true
(b) false

Practice Problems (Quiz 7)

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

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 7)

`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`:

```c
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;

```c
void const_is_shallow(const huge_record* const r) {
    int *x = r->i;
    *x = 0;
}
```

Is this a compile time error?

No.

**Practice Problems (Quiz 8)**

c : array [1..2] of integer
m : integer

```c
procedure R(k, j : integer)
    k := k+1
    j := j+2

/* begin main */
c[1] := 1
m := 1
R(m, c[m])
write m, c[m]
/* end main */
```

**Practice Problems (Quiz 9)**

class Account {
    int amount = 0;
    void deposit(int x) {
        amount = amount + 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();
        pool.awaitTermination(60, TimeUnit.SECONDS);
    }
}

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

**Practice Problems (Quiz 9)**

Generic Java class `MyList` is defined as:

```java
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)**

```java
YOUR RCSID: AND NAME:
Programming Languages Quiz 9
Monday May 2, 2016
8pointstotal
Questions 1 and 2 refer to the Java code below:
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();
        pool.awaitTermination(60, TimeUnit.SECONDS);
    }
}

Question 1. (2pts) What are all possible values for `act.ammount` at the ends?

Question 2. (1pts) Instance field `ammount` in `Account` has

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

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

Spring 16 CSCI 4430, A Milanova
Programming Language Syntax

- 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
  - CFSM and SLR(1) parsing tables
  - Conflicts in SLR(1)

PL Syntax Problems

- Problem 1. Given grammar
  \[ \text{Start} \rightarrow S \]
  \[ S \rightarrow TS \mid [ S ] S \mid \epsilon \]
  \[ T \rightarrow ( X ) \]
  \[ X \rightarrow TX \mid [ X ] X \mid \epsilon \]

  Fill in the FIRST and FOLLOW table:

<table>
<thead>
<tr>
<th>FIRST</th>
<th>FOLLOW</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

  Is this grammar an LL(1) grammar?

PL Syntax Problems

- Problem 2. Augmented grammar
  \[ S \rightarrow A \]
  \[ A \rightarrow A + A \mid B ++ \]
  \[ B \rightarrow y \]

  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 binding. With shallow binding
  - Scoping when functions are first-class values

Programming Language Semantics

- Attribute grammars
  - Attributes
    - Synthesized vs. inherited
  - Semantic rules
  - S-attributed grammars
  - L-attributed grammars

Attribute Grammar Problems

- Problem 1
  Given CFG
  \[ E \rightarrow E + T \mid T \]
  \[ T \rightarrow T \ast F \mid F \]
  \[ F \rightarrow ( E ) \mid i d \]

  Write an attribute grammar which computes at the root of the tree a \textbf{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, `(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 __________________________
                      __________________________
                      __________________________
                      __________________________
                      __________________________
                      __________________________
                      __________________________
                      __________________________
  )
  ```

More Scheme Problems

- Other problems
  - Problems 7 and 8 from practice final
  - Part 1 from practice problem set

Lambda Calculus

- Syntax and semantics
- Free and bound variables
- Rules of lambda calculus
  - α-conversion
  - β-reduction
- Evaluation order
  - Applicative order reduction
  - Normal order reduction
**Lambda Calculus Problems**

- Problem 1 (from practice final). Reduce the following term until no more reductions are possible:
  \((\lambda x. x x) ((\lambda y. y) (\lambda z. z))\)

- Problem 2 (from practice final). Suppose we define the following combinators:
  \(T = \lambda x. x x\) /* T stands for true */
  \(F = \lambda x. x y y\) /* F stands for false */
  \(N = \lambda x. (x F T)\) /* N stands for not */

  Show that \(N (N z) \Rightarrow_\beta z\) where \(z\) is either \(T\) or \(F\)

---

**Definitions**

- What is a closure, a higher-order function, tail recursion?
- What is normal-order reduction, applicative-order reduction?
- What does it mean functions are first-class values?
- Others…

---

**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 HW7, slightly modified). Consider the following type declaration in C:
  `double (*FOO(double (*)(double, double[]), double));`

  Using appropriate type constructors (pointerTo, array and \(\to\)), show the type tree for FOO

---

**Types Problems**

- Other sample problems
  - From HW7
  - Problem 6 from practice final
  - Problems from practice problem sheet
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 result
- 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 Problems

```
x : int
c : array [0..1] of int
procedure p(y, z : int)
  begin
    y := y + 1
    print x, z
  end
x := 0;
c[0] := 0; c[1] := 1;
p(x, c[x]);
print x;
```

Parameter Problems

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

- Note: All questions on parameter passing assume a language that uses the 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
  - Other…

- “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
  - Problems 7 and 10 from practice final

THANK YOU!