CSE 12 – Basic Data Structures

Prof. Christine Alvarado

[Slides borrowed/adapted from slides by Cynthia Lee, Rakesh Varna, & Roshni Chandrashekhar]
BEGINNER’S PROGRAMMING COMPETITION

SPONSORED BY BITTORRENT

SATURDAY, MAY 24TH
3PM-7:30PM AT CSE B230

JAVA-BASED, TEAMS OF 2
PIZZA PROVIDED FOR ALL CONTESTANTS
PRIZES FOR TOP CONTESTANTS

SIGN UP: WIC.UCSD.EDU/COMPETITION.HTML
Announcements

1. HW8
   1. Released Wednesday
   2. Due Tuesday week 10 (2 week assignment)
   3. Worth 400 points.
   4. It will be LONG. If you wait to start you will not succeed and you will significantly drop your grade in the class!

2. HW8 is PARTNER ENCOURAGED
   1. You must sign up via the Google form no later than THIS FRIDAY at 5pm
   2. Once you sign up you are “married” to your partner. Divorces must be done in person by me or Dr. P
Announcements

1. I am traveling this afternoon-Thursday
   1. No office hours tomorrow. If you want to meet me send me an email.
   2. No class Wednesday. Watch the video. Really, I mean it. (Did I mention HW8 is LONG?)
The pseudo code for Problem 2 should be:

```java
traversal(node) {
    if (node != null) {
        traversal(node.left)
        traversal(node.right)
        print this node
    }
}
```
Today’s topics

- Language Specification
- BNF Grammars
- Derivations, parse trees and abstract syntax trees
The Java language

Welcome to DrJava. Working directory is ...
> int x = 7;
> int y = 19;
> x + y
26
> String s = "a string";
> s + y
"a string19"
> s / y
Static Error: Bad type in numeric expression
The Unicalc language

> java Unicalc

input> 14 m + 9 m
Result: 23.0 meter

input> 60 Hz * 30 s
Result: 1800.0 Hz s

input> # 60 Hz * 30 s
Result: 1800.0

input> # 364.4 smoot
Result: 364.4 smoot

input> def smoot 67 in
Result: 67.0 in

input> # 364.4 smoot
Result: 620.13592 meter
Compilers and interpreters
Compilers and Interpreters

- One of the most important kinds of programs are programs that read other programs and translate them into action
  - Compilers and interpreters
- They both have two main steps in their processing
  - Syntax analyzer
  - Semantic evaluator
Compilers and Interpreters

- Syntax analyzer:
  - Recursively builds up an AST
- Semantic evaluator:
  - Takes an AST and does the calculation by traversing the tree
Stages of an Interpreter

Lexical Analyzer  AKA  Tokenizer

Parser

Evaluator

Environment

Printer
Stages of an Interpreter: Minimath

- **Tokenizer**
  - "5 + 3 * 2"

- **Parser**
  - 5 + 3 * 2

- **Evaluator**
  - symbol: +
    - number: 5
  - symbol: *
    - number: 3
    - number: 2

- **Environment**

- **Printer**
  - number: 11.0

"11.0"
Your own language...

Tokenizer

Parser

Evaluator

Printer

String input

List<String> tokens

AST parseTree

Quantity result

result.toString

Printed result
Getting the Symbols

String input

Tokenizer

List<String> tokens

"5+3*2" → ["5", "+", "3", "+", "2"]

"5plus3times2" → ["5", "plus", "3", "times", "2"]

"fiveplusthreetimestwo" → ["five", "plus", "three", "times", "two"]
public static void main(String[] arg) {
    int x = 32;
    int y = 9;
    int z;
    z = x+++y;
    System.out.println("x is "+ x); 33
    System.out.println("y is "+ y); 9
    System.out.println("z is "+ z); 41
}
Syntax Analyzer (Parser)

Building Abstract Syntax Trees
Understanding a sentence

He gave her food

He | gave | food

her
Understanding a sentence

He gave her cat food

BAD for Java and other PLs

He gave food

He gave food
Understanding a sentence?

Food cat gave he.
Understanding a sentence

"5 + 3 * 2"

?  
(5 + 3) * 2  
5 + (3 * 2)

"5 2 + * 3"

error
A sample grammar

\[
\begin{align*}
\langle S \rangle & := \langle S \rangle + \langle P \rangle \mid \langle S \rangle - \langle P \rangle \mid \langle P \rangle \\
\langle P \rangle & := \langle P \rangle * \langle M \rangle \mid \langle P \rangle / \langle M \rangle \mid \langle M \rangle \\
\langle M \rangle & := \langle \text{const} \rangle \mid (\langle S \rangle) \\
\langle \text{const} \rangle & := 0 \mid 1 \mid 2 \mid 3 \mid 4
\end{align*}
\]
Sample derivation

```
"2"

<s> => <p> => <m> => <const> => 2
```
You turn!

“2 + 5 + 3”

Which of the following steps can NOT follow immediately after the other in a derivation?

A. \[ \langle S \rangle \Rightarrow \langle S \rangle + \langle P \rangle \Rightarrow \]

B. \[ \langle S \rangle + \langle P \rangle + 3 \Rightarrow \langle P \rangle + \langle P \rangle + 3 \Rightarrow \]

C. \[ \langle M \rangle + \langle P \rangle + 3 \Rightarrow \langle \text{const} \rangle + \langle M \rangle + 3 \Rightarrow \]

D. \[ 2 + \langle \text{const} \rangle + 3 \Rightarrow 2 + 5 + 3 \]

E. Other/none/more
Solution

"2 + 5 + 3"

\[ <S> := <S> + <P> | <S> - <P> | <P> \]
\[ <P> := <P> * <M> | <P> / <M> | <M> \]
\[ <M> := <\text{const}> | (<S>) \]
\[ <\text{const}> := 0 | 1 | 2 | 3 | 4 \]

\[ <S> => <S> + <P> => <S> + <M> => <S> + <\text{const}> => <S> + 3 => <S> + <P> + 3 => <P> + <P> + 3 => <M> + <P> + 3 => <M> + <M> + 3 => <\text{const}> + <M> + 3 => <\text{const}> + <\text{const}> + 3 => 2 + <\text{const}> + 3 \]

\[ 2 + 5 + 3 \]
Your turn!

“2 + 5 * 3”

<\textit{S}> := <\textit{S}> + <\textit{P}> | <\textit{S}> - <\textit{P}> | <\textit{P}>
<\textit{P}> := <\textit{P}> * <\textit{M}> | <\textit{P}> / <\textit{M}> | <\textit{M}>
<\textit{M}> := <\text{const}> | (<\textit{S}>)
<\text{const}> := 0 | 1 | 2 | 3 | 4