CS 251 Fall 2015

Project 2

Tasks for Part 1

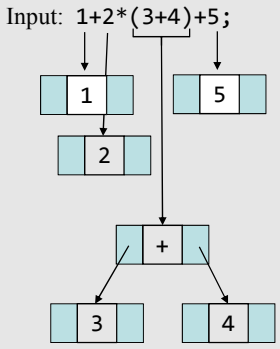
1. Build an expression tree:
   - Complete the recursive parser to build the tree; extend it to include the additional operators
   - Print it in the required format
2. Evaluate the expression tree
   - Implement a post-order tree traversal
   - Print the value in the required format

2

Parser, pseudo code

```
Tree Primary(token stream){
   if(current token == '('){
      get next token;
      Tree E1 = Expression(token stream);
      current token should be ')'

      get next token;
      return E1;
      }
   else if(current token is integer or float)
      Tree arg = new Tree(current token);
      get next token;
      return arg;
      }
   something is wrong; we quit
}
```

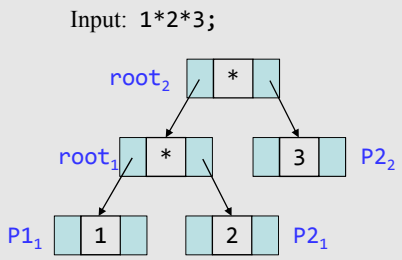


3

Parser, pseudo code

```
Tree Factor(token stream){
   Tree P1 = Primary(token stream);
   while (current token is '*' or '/'){
      Tree root = new Tree(current token);
      get next token;
      Tree P2 = Primary(token stream);
      hook up P1 and P2 with root;
      P1 = root;
   }
   return P1;
}

Tree Expression(token stream) {
   Tree F1 = Factor(token stream);
   while (current token is '+' or '-'){
      just like Factor…
   }
   return F1;
}
```



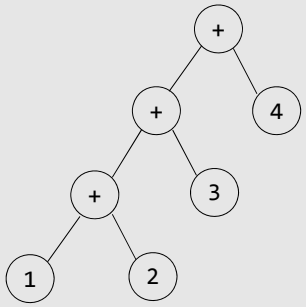
4

Associativity

All four operators associate left-to-right:

1+2+3+4 ≡ ((1+2)+3)+4

Built in with the while loops



5

Precedence

Multiply and divide take precedence over add and subtract

- Multiply and divide have equal precedence
- Add and subtract have equal precedence

Built in by looking for '*' before looking for '+' on account of the call sequence

```
1*2+3*4 ≡ (1*2)+(3*4)
1+2-3+4 ≡ (((1+2)-3)+4)
```

6

Flow

Expression calls Factor, Factor calls Primary

Primary return a tree (constant or subexpression)

Factor checks for '*'; if present builds a multiplication tree, otherwise returns to Expression

Expression checks for '+'; if present, builds an addition tree, otherwise returns

7