Question 1
• $S \rightarrow (S) S \mid \epsilon$

Question 3
\(-1 + 2 \times 3.0^{4.7} / 6\)

Lexical Analysis
• What is its purpose?
• What is the difference in a token vs. lexeme?
• Example:
  – The Brown Fox
  – if (i != 32) then j := 12
• Are spaces important?
What’s new in this grammar?

expr → expr + term { print('+') }
| expr - term { print('-') }
| term

term → term * factor { print('*') }
| term / factor { print('/') }
| factor

factor → ( expr ) { print(num.value) }
| num { print(num.value) }
| id { print(id.lexeme) }

The Scanner

for ( ; ; peek = next input character ) {
    if ( peek is a blank or a tab ) do nothing;
    else if ( peek is a newline ) line = line+1;
    else break;
}

Reading Ahead

- Read the next char, it is an “i”
- Could be int, if, or an identifier, so read next char, “f”
- Could be if, could still be an identifier, so read next char, “l”
- Oops, we’ve gone too far, push back “(“
Buffers

• Why is this important?
• Ways to implement:
  – Two pointers into buffer (start_char, lookAhead)
  – Push back buffer (peek)

The Lexical Analyzer

```java
if (peek holds a digit) {
    v = 0;
    do {
        v = v * 10 + integer value of digit peek;
        peek = next input character;
    } while (peek holds a digit);
    return token (num, v);
}
```

Figure 2.30: Grouping digits into integers

Tokens

```java
public class Token {
    public final int tag;
    public Token(int t) { tag = t; }
}
```

Figure 2.32: Class Token and subclasses Num and Word
Numbers vs. Words

1. package lexer;                  // File Num.java
2. public class Num extends Token {
3.     public final int value;
4.     public Num(int v) { super(Tag.NUM); value = v; }
5. }

1. package lexer;                  // File Word.java
2. public class Word extends Token {
3.     public final String lexeme;
4.     public Word(int t, String a) {
5.         super(t); lexeme = new String(a);
6.     }
7. }

Figure 2.33: Subclasses Num and Word of Token