All students should read textbook chapter 3 through to §3.3.

Distribute the following questions across the members of your group. You will share your solutions (and most importantly the *method* of your solutions) during the next lecture period. Divide up the questions so that **each** question has at least two solutions from different group members.

1. Page 106, question 1
2. Consider the Java snippet below, what token stream would be generated by the scanner? Which tokens would need source values associated with them and which do not?

   Your answer does not need to be specific to Java (I don't expect you look research the inner workings of `javac`). Your answer should be understandable by someone knowledgeable in compilers and lexers, but not the Java language per se.

   Hint: Before tackling this question review `show_overview.pdf` so that you have a more concrete idea of **what is passed from the lexing stage to the parser (grammar analysis stage)** during compilation — this is the token stream!

```
public static boolean[][] nextDay(boolean[][] world){
        boolean[][] newWorld
                = new boolean[world.length][world[0].length];
        int num, r=1;
        for(int c = 0; c < world[0].length; c++){
                num = numZombies(world, r, c);
                if( walkingDeadRNG(num, world[r][c]) )
                        newWorld[r][c] = true;
        }
        return newWorld;
}
```

3. Page 106, question 2
4. Using the notation for regular expressions given in §3.2, write regular expressions for:
   i. A real number ($\Re$) written in conventional programmatic forms, including scientific notation with `E` or `e` preceding the exponent.
   ii. A C or C++ identifier with exactly three underscores in its value.
   iii. . . . a C or C++ identifier with either one or two underscores in its value.
   iv. A conventional Internet Email address.[1]

[1]Unconventional yet technically valid Email addresses are tricky business — stick to the common pattern you know so well.