CS241e
Review Solutions

1. Tokens

2. Scanning, Parsing, Type Checking, Code Generation, Assembling

3. False, notice that we have to type-check in Lacs, so it is not a context-free language (i.e. it can not be determined by a context-free grammar)

4. True, notice that any DFA with n states that accepts a string of length n must have a cycle, and a string of transitions from the cycle to an accept state. Since we can loop through the cycle an arbitrary number of times before going to the accept state, there are an infinite number of string accepted by the DFA.

5. False, the set $\{cat, dog\}$ can be recognized by 2 distinct DFAs, one where the accept state for both paths is the same state and one where they are unique states.

6. True, consider the NFA with 4 states (1,2,3,4), where 1 is the start state and 4 is the accept state. a goes from 1 to 2 and 3, b goes from 1 to 3, b goes from 3 to 2, and b goes from 2 to 4. This NFA accepts the set $\{abb, ab, bbb\}$, however there is no DFA that recognizes that set with 4 states.

7. Notice that the string "xyy" can be created through 2 distinct parse trees:
   S can derive AB, and the A derives AB (getting us the string ABB which dervies xyy), or S derives AB, and then the B revies BB (getting us the string ABB which derives xyy).

8. $S \rightarrow (S)S$
   $S \rightarrow \epsilon$

9. Notice that the set of balanced parentheses is not regular, since we would need an infinite number of states to keep track of an arbitrary number of opening brackets, thus the set of balanced parentheses is context-free but not regular.

10. Yes, we can construct the following CFG to recognize the set of palindromes over binary strings:
    $S \rightarrow 0S0$
    $S \rightarrow 1S1$
    $S \rightarrow 1$
    $S \rightarrow 0$
    $S \rightarrow \epsilon$

11. The main difference between a closure and a "normal" function is that a closure has access to an external environment which a regular function does not have. In terms of implementation, the environment of a closure must be stored on the heap.

12. See above, the main reason that you cannot do this is that a closure behaves differently than a normal function

13. Parsing, there is not a valid rule in the grammar that can give us the expression a++b.

14. Scanning, "$" is not a valid token, nor does any valid token in Lacs start with "$"

15. Type Checking, the function will pass scanning and parsing, but the function "main" should return a function whereas here it returns an Int.

16. Fragmentation is a problem in memory where we have a bunch of small unused blocks of memory inbetween large blocks of memory which cannot be used since we must give requested memory in contiguous sections. Using this method of memory would solve that problem. An issue that occurs is that if very small amounts of memory are requested, we are wasting a lot of space since there is memory within the blocks that we are not using.

17. The leftmost two functions are simply the identity function, so this reduces down to the function $\lambda a.\lambda b.a$, which is a function which takes in two arguments and returns the first argument. This is also defined simply as "TRUE" in the lambda calculus.

18. We get the tokens "BABABBAB" and "B".