CFL AND PDA PROBLEMS

* (1) Stacks

a) Suppose you are given an empty stack. What does it look like after the following operations?

Push a.
Push b.
Push c.
Pop.
Pop.
Push a.
Push b.
Pop.
Push b.
Push c.
Push c.
Pop.
Pop.

b) Suppose again you start with an empty stack and you’re given a list of pushes and pops (like in part a). How can you determine how many elements will be in the stack at the end without actually doing the operations? (Ask yourself first: How can you tell whether the stack will be empty, i.e. have zero elements?) Assume that you never have to try to pop an empty stack.

* (2) PDAs Like to Read

We said in class that PDAs can look at the very top symbol on the stack using the PEEK operations. What if we got rid of the PEEK operations and only kept PUSH and POP for the PDA stack: would your PDA still be able to read the top somehow? If yes, how? If not, why?
** (3) Making PDAs

Give a PDA that recognizes each of the following languages as by both describing an NFA-with-a-stack machine and by writing an informal English description of the PDA.

a) The language \{a^n b^n \mid n \geq 0\}

b) The complement of the language \{a^n b^n \mid n \geq 0\}
(continued)

c) The set of strings over the alphabet \{a, b\} with more a’s than b’s.

d) \{x_1\#x_2\#x_3\#\ldots\#x_k \mid k \geq 1, \ \Sigma = \{a, b\}, \text{ and at least two of the } x \text{’s are reverses of each other}\}
** (4) Ambiguous Grammar: Why Dating is Confusing

*(From Sipser, Introduction to the Theory of Computation, 2nd ed., section 2.1)*

Show that the string

```
the girl touches the boy with the flower
```

... has two different leftmost derivations in the grammar $G_2$ on your handout (Sipser, p 101). Describe in English, or draw pictures for, the two different real-world meanings of the string.

**/**** (5) Ambiguous Programming Language Grammar

*(From Sipser, Introduction to the Theory of Computation, 2nd ed., Problem 2.27)*

Let $G$ be the following grammar:

- $<\text{STMT}> \rightarrow <\text{ASSIGN}> | <\text{IF-THEN}> | <\text{IF-THEN-ELSE}>$
- $<\text{IF-THEN}> \rightarrow \text{if } \text{condition} \text{ then } <\text{STMT}>$
- $<\text{IF-THEN-ELSE}> \rightarrow \text{if } \text{condition} \text{ then } <\text{STMT}> \text{ else } <\text{STMT}>$
- $<\text{ASSIGN}> \rightarrow a:=1$

Set of terminals $\Sigma = \{\text{if}, \text{condition}, \text{then}, \text{else}, a:=1\}$
Set of variables $V = \{<\text{STMT}>, <\text{IF-THEN}>, <\text{IF-THEN-ELSE}>, <\text{ASSIGN}>\}$

$G$ is a natural-looking grammar for a fragment of a programming language. But $G$ is ambiguous.

** a) Show that $G$ is ambiguous. (Hint: Give a string generated by $G$ and show you can derive it two different ways by drawing two different parse trees for it.)
**** b) Give a new unambiguous grammar for the same language: it should recognize exactly the same strings, but not have any possibility of ambiguous derivations.

**** (6) CFL Closure Under Regular Operations

Context-free languages are closed under the regular operations (union, concatenation, star), but not under complementation or intersection.

a) Prove CFLs are closed under union.
   (Hint: Start with two CFLs, $L_1$ and $L_2$, and show that their union must be a CFL too.)
(continued)

b) Prove CFLs are closed under concatenation.

c) Prove CFLs are closed under star.