1. *(5 points)* Use the Pumping Lemma for Regular Languages to prove that the following languages are **not** regular. (Remember that any pumping lemma proof follows a similar pattern to the one given in class, starting with, "Suppose that L **is** regular. Then by the pumping lemma, there is an integer K such that if w is any string $w \in L$ with $|w| \geq K$, then w can be written $w = xyz$ where $|xy| \leq K$, $|y| \geq 1$, and $xy^nz \in L$ for all natural numbers n. But let $w = \dots$, which is in L and $|w| \geq K$.")

a) $L = \{ww \mid w \in \{a,b\}^*\}$. L consists of strings of a's and b's where the first half of the string is identical to the second half, such as $abbababbab$.

b) $L = \{w \in \{a,b\}^* \mid n_a(w) < n_b(w)\}$, where $n_\sigma(w)$ means the number of σ's in w.

Answer:

a) Let $L = \{ww \mid w \in \{a,b\}^*\}$. Suppose, for the sake of contradiction, that L **is** regular. By the Pumping Lemma, there is a $K \in \mathbb{N}$ such that if w is any string in L with $|w| \geq K$, then w can be written $w = xyz$ where $|xy| \leq K$, $|y| \geq 1$, and $xy^nz \in L$ for all natural numbers n. But let $w = a^Kba^Kb$, which is in L and $|w| \geq K$. Write $w = xyz$, as in the Pumping Lemma. Since $|xy| \leq K$, and the first K characters in w are a's, we see that y must consist entirely of a's. So $y = a^j$ for some $j > 0$. By the Pumping Lemma, $xy^2z \in L$. Now $xy^2z = a^{K+j}ba^Kb$, where $j > 0$. Since there are more a's in the first group of a's than in the second group, $xy^2z \notin L$. This contradicts $xy^2z \in L$, and this contradiction proves that L cannot be regular.

b) Let $L = \{w \in \{a,b\}^* \mid n_a(w) < n_b(w)\}$. Suppose, for the sake of contradiction, that L **is** regular. By the Pumping Lemma, there is a $K \in \mathbb{N}$ such that if w is any string in L with $|w| \geq K$, then w can be written $w = xyz$ where $|xy| \leq K$, $|y| \geq 1$, and $xy^nz \in L$ for all natural numbers n. But let $w = a^Kb^{K+1}$, which is in L and $|w| \geq K$. Write $w = xyz$, as in the Pumping Lemma. Since $|xy| \leq K$, and the first K characters in w are a's, we see that y must consist entirely of a's. So $y = a^j$ for some $j > 0$. By the Pumping Lemma, $xy^2z \in L$. Now $xy^2z = a^{K+j}b^{K+1}$, where $j > 0$. Since $j \geq 1$, $K + j \geq K + 1$. That is, the number of a's in xy^2z is **not** less than the number of b's in xy^2z. So $xy^2z \notin L$. This contradicts $xy^2z \in L$, and this contradiction proves that L cannot be regular.

2. *(5 points)* Consider the context free grammar shown at the right.

$$
\begin{aligned}
S &\longrightarrow TR \\
T &\longrightarrow aTb \\
T &\longrightarrow \varepsilon \\
R &\longrightarrow cRd \\
R &\longrightarrow c
\end{aligned}
$$

a) Write a derivation for the string $aabbc$ using this grammar.

b) Write a derivation for the string $abcccdd$ using this grammar.

c) Find the language generated by this grammar. Briefly justify your answer.

Answer:

a)
$$
\begin{aligned}
\mathrm{S} &\Longrightarrow TR \\
&\Longrightarrow aTbR \\
&\Longrightarrow aaTbbR \\
&\Longrightarrow aabbR \\
&\Longrightarrow aabbc
\end{aligned}
$$

b) $\mathrm{S} \Longrightarrow TR$
$\Longrightarrow aTbR$
$\Longrightarrow abR$
$\Longrightarrow abcRd$
$\Longrightarrow abccRdd$
$\Longrightarrow abcccdd$

c) This grammar generates the language $\{a^n b^n c^{m+1} d^m \mid n, m \in \mathbb{N}\}$. (This could also be written $\{a^n b^m c^k d^\ell \mid n = m \text{ and } k = \ell + 1\}$.) The only rule that applies to the start symbol S is $S \longrightarrow TR$, so any string in the language consists of a string generated from T followed by a string generated from R. From T, the rule $T \longrightarrow aTb$ can only generate the same number of a's and b's, with the T in the middle. Eventually, $T \longrightarrow \varepsilon$ must be applied to make the T go away, leaving a string $a^n b^n$ for some $n \in \mathbb{N}$. Similarly, $R \longrightarrow cRd$ always generates the same number of c's and d's. Then the rule $R \longrightarrow c$ must be applied for the R to go away, leaving a string $c^k d^\ell$ where k is $\ell + 1$.

3. (*10 points*) For each of the following languages, create a Context-Free Grammar that generates that language. **Explain in words why your grammar works**. (As a hint for part (b), think about what you need to add to the grammar that we did in class for $\{w \in \{a,b\}^* \mid n_a(w) = n_b(w)\}$. As a hint for part (d), note that letters can match in pairs a/c, a/d, b/c, and b/d.)

a) $\{a^n b a^n \mid n \in \mathbb{N}\}$

b) $\{w \in \{a,b\}^* \mid n_a(w) < n_b(w)\}$

c) $\{a^n b^m c^k d^l \mid m = k \text{ and } n = l\}$

d) $\{a^n b^m c^k d^l \mid n + m = k + l\}$

Answer:

a) $S \longrightarrow aSa$
$S \longrightarrow b$

This grammar generates equal numbers of a's on either side of S. When the rule $S \longrightarrow b$ is applied, it puts a b between the two groups of a's.

b) $T \longrightarrow SbS$
$S \longrightarrow SS$
$S \longrightarrow aSb$
$S \longrightarrow bSa$
$S \longrightarrow \varepsilon$
$S \longrightarrow b$

The start symbol for this grammar is T. Any string in the language must have at least one b that does not match any a. The first rule in the grammar must be the first rule applied in any derivation, and it adds a b with no matching a. The next four rules generate equal numbers of a's and b's while the last rule makes it possible to add more extra b's.

c) $S \longrightarrow aSd$
$S \longrightarrow T$
$T \longrightarrow bTc$
$T \longrightarrow \varepsilon$

The first rule can be used repeatedly to generate strings of the form $a^n S d^\ell$ where $n = \ell$. Eventually, the rule $S \longrightarrow T$ must be applied. Then the third rule can be applied repeatedly to generate strings of the from $a^n b^m T c^k d^\ell$, with $m = k$. The rule $T \longrightarrow \varepsilon$ is used at the end to get remove the T.

d)

$$\begin{aligned} S &\longrightarrow aSd \\ T &\longrightarrow aTc \\ R &\longrightarrow bRd \\ U &\longrightarrow bUc \\ S &\longrightarrow T \\ S &\longrightarrow R \\ T &\longrightarrow U \\ R &\longrightarrow U \\ U &\longrightarrow \varepsilon \end{aligned}$$

The first four rules can generate the four possible pairs of letters, a/c, a/d, b/c, and b/d. They ensure that the number of a's plus the number of b's must always be equal to the number of c's plus the number of d's. The next four rules ensure that the characters can only occur in the correct order. For example, once we stop generating a/d pairs with the first rule, we can change to generating either a/c or b/d pairs, inside the paired a's and d's. And we can always finish by transitioning to U to generate b/c pairs between any a's and d's. The last rule allows the U to disappear in the end.

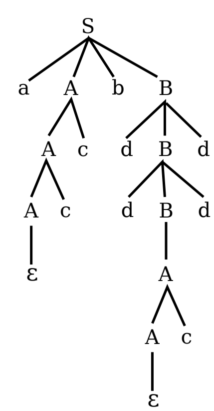
4. (*3 points*) Given the following (very incomplete) BNF grammar for "names" in Java, write down **six** "names" generated by this grammar. Your examples should demonstrate all the possibilities represented in the rules.

$$\begin{aligned} \langle \textit{name} \rangle &::= \langle \textit{object\_ref} \rangle \text{ [``.'' } \langle \textit{identifier} \rangle \text{ ]} \\ \langle \textit{object\_ref} \rangle &::= \langle \textit{identifier} \rangle \mid \langle \textit{method\_call} \rangle \\ \langle \textit{identifier} \rangle &::= \text{``a''} \mid \text{``x''} \mid \text{``y''} \mid \text{``z''} \\ \langle \textit{method\_call} \rangle &::= \langle \textit{identifier} \rangle \text{ ``('' } \langle \textit{name} \rangle \text{ [``,'' } \langle \textit{name} \rangle \text{ ]}\ldots \text{ ``)''} \end{aligned}$$

Answer:

```
x            a(x)            x.y
a(x).y       a(x,y,z)        a(x,y,z).a
a(x(y))      z(a(x).y,z)     z(y(z.a,z.a),x).a
```

5. (*7 points*) Suppose that the parse tree at the right is based on a context-free grammar G that has exactly five production rules.



a) Give the five production rules that must be part of G for this parse tree to be valid.

b) What is the yield of this parse tree, that is, the string that is being parsed?

c) Give the left derivation corresponding to this parse tree.

d) Give the right derivation corresponding to this parse tree.

e) Draw a parse tree using the same grammar for the string *abdccd*.

Answer:

a)
$$\begin{aligned} S &\longrightarrow aAbB \\ A &\longrightarrow Ac \\ A &\longrightarrow \varepsilon \\ B &\longrightarrow dBd \\ B &\longrightarrow A \end{aligned}$$

b) *accbddcdd*

c)
$$
\begin{aligned}
\text{S} &\Longrightarrow aAbB \\
&\Longrightarrow aAcbB \\
&\Longrightarrow aAccbB \\
&\Longrightarrow accbB \\
&\Longrightarrow accbdBd \\
&\Longrightarrow accbddBdd \\
&\Longrightarrow accbddAdd \\
&\Longrightarrow accbddAcdd \\
&\Longrightarrow accbddcdd
\end{aligned}
$$

d)
$$
\begin{aligned}
\text{S} &\Longrightarrow aAbB \\
&\Longrightarrow aAbdBd \\
&\Longrightarrow aAbddBdd \\
&\Longrightarrow aAbddAdd \\
&\Longrightarrow aAbddAcdd \\
&\Longrightarrow aAbddcdd \\
&\Longrightarrow aAcbddcdd \\
&\Longrightarrow aAccbddcdd \\
&\Longrightarrow accbddcdd
\end{aligned}
$$

e)