1. **(30 points):** Binary multiplication.
As in homework 2, let $\Sigma = \{0,1\}^3$, i.e. the set of tuples consisting of three bits. For, $(a,b,c) \in \Sigma$, define

$$\begin{array}{rcl} \mathsf{first}((a,b,c)) & = & a \\ \mathsf{second}((a,b,c)) & = & b \\ \mathsf{third}((a,b,c)) & = & c \end{array}$$

We overload first, second, and third to strings as shown below:

$$\begin{array}{rcl} \mathsf{first}(\epsilon) & = & \epsilon \\ \mathsf{first}(x \cdot \mathsf{c}) & = & \mathsf{first}(x) \cdot \mathsf{first}(\mathsf{c}) \\[1ex] \mathsf{second}(\epsilon) & = & \epsilon \\ \mathsf{second}(x \cdot \mathsf{c}) & = & \mathsf{second}(x) \cdot \mathsf{second}(\mathsf{c}) \\[1ex] \mathsf{third}(\epsilon) & = & \epsilon \\ \mathsf{third}(x \cdot \mathsf{c}) & = & \mathsf{third}(x) \cdot \mathsf{third}(\mathsf{c}) \end{array}$$

For example, if $s = (0,0,0)(0,0,1)(0,1,0)(0,1,1)(1,0,0)(1,0,1)$, then $\mathsf{first}(s) = 000011$, $\mathsf{second}(s) = 001100$, and $\mathsf{third}(s) = 010101$. For $s \in \{0,1\}^*$, let $\mathsf{binary}(s)$ denote the binary value of s when the most significant bit is the first symbol of the string:

$$\begin{array}{rcl} \mathsf{binary}(\epsilon) & = & 0 \\ \mathsf{binary}(\mathsf{0}) & = & 0 \\ \mathsf{binary}(\mathsf{1}) & = & 1 \\ \mathsf{binary}(x \cdot \mathsf{c}) & = & 2 * \mathsf{binary}(x) + \mathsf{binary}(c) \end{array}$$

Let B denote the language of binary multiplication:

$$B = \{w \mid \mathsf{binary}(\mathsf{third}(w)) = \mathsf{binary}(\mathsf{first}(w)) * \mathsf{binary}(\mathsf{second}(w))\}$$

(a) **(10 points):** Use the pumping lemma for regular languages to show that B is not regular.

Solution:
Rather than writing out the tuples, I'll just write strings of equal length of x, y, and z, with the interpretation that x represents the first component of the string for B; y represents the second, and z represents the third. Thus, we want $z = x * y$.
Let k be the demon's choice for a pumping lemma constant. Let $x = y = 0^k 1 0^k$, and $z = 1 0^{2k}$. Thus, x, and y encode the value 2^k and z encodes 2^{2k}, and $z = x * y$ as required. Force the demon to pump in the first k symbols of the string. This leaves pumping leaves the values of x and y unchanged but changes the value of z. Thus, pumping leads to strings where $z \neq x * y$. This shows that B is not regular.

(b) **(20 points):** Is B context-free? Prove your answer.

Solution:
This time, we need to be a little bit more subtle. Note that with our previous string, the demon could break it up to obtain $x' = y' = 0^{k-1} 0^i 1 0^i 0^{k-1}$ which encodes 2^{k+i-1} and $z' = 10k - 20^i 00^i 0^{k-1} = 10^{2(k+i-1)}$ which encodes $2^{2(k+i-1)}$. Thus, $z = x * y$.
The problem with the previous example was that it only has two fields of zeros, and the demon can pump them separately. To overcome this, let $x = y = 0^{2k+2} 1 0^k 1 0^k 1$ which encodes $2^{2(k+1)} + 2^{k+1} + 1$. Thus, $x * y$ should encode $2^{4(k+1)} + 2 * 2^{3(k+1)} + 3 * 2^{2(k+1)} + 2 * 2^{k+1} + 1$, which means that $z = 1 0^{k-1} 1 0^k 11 0^{k-1} 1 0^{k+1} 1$ (now you know why I put the number of leading zeros that I did in front of z).

Now if we let the demon pump this string, it might pick some substrings that contain both zeros and ones. While this may or may not produce a string that preserves the relationship that $z = x * y$, it will certainly make the multiplication problem more complicated – we don't want the demon to win by confusing us!
To preserve our sanity, we take advantage of the property that CFLs are closed under intersection with regular languages. In particular, we restrict x and y to be of the form $0^+100^+10^+1$. Likewise, we restrict z to be of the form $10^+10^+110^+10^+1$. Each of these restrictions can be effected by intersecting T with a regular language. Let T' be T intersected with the regular languages for these restrictions on x, y, and z. If T is context-free, then T' must be context free as well.
To be fair, we must give the demon a chance to choose an new k. I'll assume $k > 1$ to ensure that z has the desired structure – we're always allowed to present a longer string than the lower bound imposed on us by the demon. My string consists of $x = y = 0^{2k+2}10^k10^k1$ and $z = 10^{k-1}10^k110^{k-1}10^{k+1}1$ as before. We now politely ask the demon to pump (x, y, z).
Let's say the demon chooses $abcde = (x, y, z)$, with $|bd| \geq 1$ and $|bcd| \leq k$, and claims that $ab^icd^ie \in B$ for any $i \geq 0$. I'll write $x = \alpha 1 \beta 1 \gamma 1$, where $\alpha = 0^{2k+2}$, $\beta = \gamma = 0^k$. By arguments similar to those for part (a), the demon can't choose b and d to both correspond to the α part of x, as this would change the value encoded by z without changing the value for x or y. Likewise, the demon can't choose both b and d to be in β or γ, because with $i = 2$, $x, y > 2^{2k+2+|bd|}$ and $z < 2^{4k+4+bd}$ which implies $z \neq x * y$. Thus, b must correspond to a substring of α, and d must correspond to a substring of β or γ.
Consider the case where d is a substring of β. Let x' be the value for x corresponding to ab^2cd^2e, and likewise for y' and z'. We note that x' encodes the value $2^{2k+2+|d|} + 2^{k+1} + 1$. Thus the product of x_i and y_i is

$$2^{2(2k+2+|d|)} + 2 * 2^{3k+3+|d|} + 2 * 2^{2k+2+|d|} + 2^{2k+2} + 2 * 2^{k+1} + 1$$

The corresponding string has the form $0^*10^+0^+10^+10^+10^+10^+1$ which does not satisfy the restriction that we put on z. Thus, this product is not in T'. This shows that T' is not context-free. Therefore, T is not context-free.

2. **(30 points):** Let $\Sigma = \{a, b\}$. As usual, let $\#\mathsf{a}(x)$ denote the number of occurences of a in x, and let $\#\mathsf{b}(x)$ denote the number of occurences of b. Show that each of the languages below is context-free:

   (a) **(10 points)**: $\{x | \ \#\mathsf{a}(x) \leq \#\mathsf{b}(x)\}$

   Solution:
   In the solution to HW3, we described a CFG for the language where $\#\mathsf{a}(x) = \#\mathsf{b}(x)$. We modify this grammar to recognize this language:

$$\begin{array}{rcl} S & \rightarrow & EaE \mid SS \\ E & \rightarrow & \epsilon \mid aEb \mid bEa \mid EE \end{array}$$

   The non-terminal E generates strings with an equal number of a and b symbols. Thus, EaE generates strings with one more a than b. Finally, the production $S \rightarrow SS$ allows the generation of strings with more than one excess a.

(b) **(10 points)**: $\left\{x \,\middle|\, |\#\mathsf{a}(x) - \#\mathsf{b}(x)| < 3\right\}$

Solution:

This time, I'll modifify the PDA from my solution to HW3:

$$\begin{aligned}
M &= (\{q_1\}, \{a, b\}, \{\bot, a, b\}, \delta, q_1, \bot, \emptyset)\\
\delta &= \{\ (q_1, a, \bot) \rightarrow (q_1, a\ \bot)\\
&\qquad (q_1, a, a) \rightarrow (q_1, aa)\\
&\qquad (q_1, a, b) \rightarrow (q_1, \epsilon)\\
&\qquad (q_1, b, \bot) \rightarrow (q_1, b\ \bot)\\
&\qquad (q_1, b, b) \rightarrow (q_1, bb)\\
&\qquad (q_1, b, a) \rightarrow (q_1, \epsilon)\\
&\qquad (q_1, \epsilon, \bot) \rightarrow (q_1, \epsilon)\\
&\quad \}
\end{aligned}$$

To accept if $|\#\mathsf{a}(x) - \#\mathsf{b}(x)| < 3$, we can pop up to two a or b symbols off the stack after reading the input string. I'll do this by adding two more states, q_2, and q_3. Furthermore, I'll add the transitions

$$\begin{array}{rcl}
(q_1, \epsilon, a) & \rightarrow & (q_2, \epsilon)\\
(q_1, \epsilon, b) & \rightarrow & (q_2, \epsilon)\\
(q_2, \epsilon, \bot) & \rightarrow & (q_1, \epsilon)\\
(q_2, \epsilon, a) & \rightarrow & (q_3, \epsilon)\\
(q_2, \epsilon, b) & \rightarrow & (q_3, \epsilon)\\
(q_3, \epsilon, \bot) & \rightarrow & (q_1, \epsilon)
\end{array}$$

(c) **(10 points)**: $\left\{x \,\middle|\, (|\#\mathsf{a}(x) - \#\mathsf{b}(x)| \bmod 3) = 2\right\}$

Solution:

In this case, I want to make sure that after reading the input string, that the number of symbols on the stack has a remainder of three when divided by three. I'll start with the machine from my solution to HW3 as described above and remove the transition

$$(q_1, \epsilon, \bot) \rightarrow (q_1, \epsilon)$$

I'll add two states, q_2, and q_3 and the transitions:

$$\begin{array}{rcl}
(q_1, \epsilon, a) & \rightarrow & (q_2, \epsilon)\\
(q_1, \epsilon, b) & \rightarrow & (q_2, \epsilon)\\
(q_2, \epsilon, a) & \rightarrow & (q_3, \epsilon)\\
(q_2, \epsilon, b) & \rightarrow & (q_3, \epsilon)\\
(q_3, \epsilon, a) & \rightarrow & (q_1, \epsilon)\\
(q_3, \epsilon, b) & \rightarrow & (q_1, \epsilon)
\end{array}$$

This machine just pops symbols off the stack. The number of a's or b's left on the stack initially after reading the input has a remainder of two when divided by three iff the machine uncovers the $\bot$ symbol when moving to state q_3. Thus, I add one more transition:

$$(q_3, \epsilon, \bot) \rightarrow (q_3, \epsilon)$$

This machine accepts he specified language on emptyh stack.

3. **(20 points):** (Kozen, Miscellaneous Exercise 31)
Let $\Sigma = \{a, b, \$\}$. One of the two sets is regular and the other is not. Which is which? Prove your answers.

   (a) **(10 points)**: $\{w \mid \exists x, y \in \{a, b\}^*.\ (w = xy) \wedge (\#\mathsf{a}(x) = \#\mathsf{b}(y))\}$

   Solution:

   This language is regular. In fact, it is the same as Σ^*.
   Let $n = |w|$. For $0 \leq i \leq n$, let x_i be the first i symbols of w, and let y_i be the last $n - i$ symbols. Thus, $w = x_i y_i$. We note that

$$\begin{array}{rcl} \#\mathsf{a}(x_0) - \#\mathsf{b}(y_0) & = & 0 - \#\mathsf{b}(w) \\ & = & -\#\mathsf{b}(w) \end{array}$$

   Likewise, $\#\mathsf{a}(x_n) - \#\mathsf{b}(y_n) = \#\mathsf{a}(w)$. Furthermore,

$$\#\mathsf{a}(x_{i+1}) - \#\mathsf{b}(y_{i+1}) \quad = \quad (\#\mathsf{a}(x_i) - \#\mathsf{b}(y_i)) + 1$$

   because with each step, either $\#\mathsf{a}(x_i)$ increases by one or $\#\mathsf{b}(y_i)$ decreases by one. Thus, $\#\mathsf{a}(x_{\#\mathsf{b}(w)}) = \#\mathsf{b}(y_{\#\mathsf{b}(w)})$. Therefore, w is in the language. The choice of w was arbitrary. Thus, this language contains all strings; it is Σ^*.

   (b) **(10 points)**: $\{w \mid \exists x, y \in \{a, b\}^*.\ (w = x\$y) \wedge (\#\mathsf{a}(x) = \#\mathsf{b}(y))\}$

   Solution:

   This language is not regular. Let the demon chose k, and let $w = a^k\$b^k$. Clearly, w is in this language (e.g. let $x = a^k$, and let $y = \$b^k$). Now, force the demon to pump the a^k prefix. This will change the number of a's preceding the \$, but the number of b's will be unchanged. This produces a string that is not in the language. Therefore, this language is not context free.

4. **(20 points):** (Kozen, Miscellaneous Exercise 75)
Define a context-free grammar for regular expressions over an alphabet Σ. Your grammar should have the terminal symbols $\Sigma \cup \{\epsilon, \phi, \cdot, +, (,), ^*\}$. Your grammar should be unambiguous.

Solution:

$$\begin{array}{rcll} S & \rightarrow & \mathit{expr} & \\ \mathit{expr} & \rightarrow & \mathit{term} \parallel \mathit{expr} + \mathit{expr} & \\ \mathit{term} & \rightarrow & \mathit{factor} \parallel \mathit{term} \cdot \mathit{factor} & \\ \mathit{factor} & \rightarrow & \mathit{kangaroo} \parallel \mathit{kangaroo}^* & \\ \mathit{kangaroo} & \rightarrow & \mathbf{epsilon} \parallel \mathbf{phi} \mid (\ \mathit{expr}\ ) & \\ \mathit{kangaroo} & \rightarrow & \mathsf{c}, & \text{for each } \mathsf{c} \in \Sigma \end{array}$$