# HW Solution

**CS/ECE 374: Algorithms & Models of Computation, Spring 2019** Version: **1.0**

Submitted by:

- ≪**Alan Lee**≫: ≪**alanlee2**≫
- ≪**Joshua Burke**≫: ≪**joshuab3**≫
- ≪**Gerald Kozel**≫: ≪**gjkozel2**≫

(100 PTS.) Grammar it.

Describe a context free grammar for the following languages. Clearly explain how they work and the role of each non-terminal. Unclear grammars will receive little to no credit.

**1** (40 PTS.) $\left\{a^i b^j c^k d^\ell e^t \;\middle|\; i,j,k,\ell,t \geq 0 \text{ and } i+j+k+t=\ell\right\}$.

**2** (60 PTS.) (Harder.) $L = \{z \in \{a,b,c\}^* \mid$ there is a suffix $y$ of $z$ s.t. $\#_a(y) > \#_b(y)\}$.

(Hint: First solve for the case that $z$ has no $c$s.)

## 11 Solution:

1. $G = (V, T, P, S)$

   $V = \{S, A, B, C, E\}$

   $T = \{a, b, c, d, e\}$

   $P = \{$

   $S \to AE$

   $A \to \epsilon \mid aAd \mid B$

   $B \to \epsilon \mid bBd \mid C$

   $C \to \epsilon \mid cCd$

   $E \to \epsilon \mid dEe$

   $\}$

   This grammar builds builds the string by building the suffix and prefix each from the outside inwards, recursively. Starting with S, we generate non-terminals for the beginning and end of the string: A and E. E can generate a suffix of $t$ d's followed by $t$ e's.

   A generates the prefix from the outside in, first adding equal numbers of a's and d's to the beginning and end of the prefix, then it will turn into B which can generate equal numbers of b's and d's inside of that, then it will turn into C which does the same but for c's and d's. C can only do this or turn into the empty string.

2. $G = (V, T, P, S)$

   $V = \{S, A, B\}$

   $T = \{a, b, c\}$

   $P = \{$

   $S \to A$

   $A \to aA \mid cA \mid bA \mid aB$

   $B \to \epsilon \mid aBb \mid cB \mid Bc$

   $\}$

   We though about this question similar to the ideas posted in Piazza post @412. There are 3 major parts of any string within language L. The first part can be any permutation of valid characters a, b, c (represented by the first 3 transitions of non-terminal A). The second part is a single 'a' character (represented by the last transition of non-terminal A). Lastly, the third part is any string with an equal amount of a and b characters (represented by non-terminal B). combining these parts in this sequence, we ensure that there exists a suffix with more a's than b's.