Problem 1 (10 pt) PCFGs and HMMs

a.

$$\Pr_{joint} = \Pr(left\ tree) + \Pr(right\ tree) = 0.0072 + 0.006 = 0.0132$$

b.

$\Pr(left\ sequence) = (Prep \mid Start) * (they \mid prep) * (V \mid Prep) * (are \mid V) * (Adj \mid V) * (baking \mid Adj) * (N \mid Adj) * (potatoes \mid N) = 0.1 * 1.0 * 0.8 * 0.5 * 0.3 * 1 * .6 * 1 = .0072$

$\Pr(Right\ sequence) = (Prep \mid Start) * (they \mid prep) * (Aux \mid Prep) * (are \mid Aux) * (V \mid aux) * (baking \mid V) * (N \mid V) * (potatoes \mid N) = 0.1 * 1 * 0.2 * 1 * 1 * 0.5 * 0.6 * 1 = 0.006$

Transition Probabilities:
Pr(Prep | Start) = 0.1
Pr(Aux | Prep) = 0.2
Pr(V | Prep) = 0.8
Pr(V | Aux) = 1
Pr(N | V) = 0.6
Pr(Adj | V) = 0.3
Pr(N | Adj) = 0.6

Emission Probabilities:
Pr(they | prep) = 1
Pr(are | V) = 0.5
Pr(baking | adj) = 1
Pr(potatoes | N) = 1
Pr(are | aux) = 1
Pr(baking | V) = 0.5

(other transition probabilities used to fill out and sum to 1 for valid HMM)
Pr(V | Start) = 0.9
Pr(Aux | V) = 0.1
Pr(aux | adj) = 0.4

c.
In general it is not possible to translate any PCFG into an HMM that produces the same joint probability as the PCFG. It is not possible because the size of the regular languages that can be represented by regular grammar as finite state automata (HMMs) is strictly smaller. A CFG with multiple deeply recursive and parse trees we may not be able to get the same joint probability when trying to translate to HMM.

Problem 2

Problem 2	10 pts	

They are baking potatoes

Chart 0

S_0 S → · NP VP [0,0] init

S_1 NP → · Adj NP [0,0] predict S_0

S_2 NP → · PREP [0,0] predict S_0

S_3 NP → · N [0,0] predict S_0

S_4 Adj → · baking [0,0] predict S_1

S_5 PRP → · they [0,0] predict S_2

S_6 N → · potato [0,0] predict S_3

Chart 1

S_7 PRP → they · [0,1] Scan S_5

S_8 NP → PRP · [0,1] completing S_2 w/ S_7

S_9 S → NP · VP [0,1] completing S_0 w/ S_8

S_{10} VP → · VP NP [1,1] predict S_9

S_{11} VP → · Aux V NP [1,1] predicti S_9

S_{12} V → · baking [1,1] predict S_{10}

S_{13} V → · Are [1,1] predict S_{10}

S_{14} Aux → · Are [1,1] predict S_{11}

Chart 2

S_{15} V → are · [1,2] scan S13

S_{16} Aux → are · [1,2] scan S_{14}

S_{17} VP → V · NP [1,2] complete S_{10} w/ S_{15}

S_{18} VP → Aux · V NP [1,2] complete S_{11} w/ S_{16}

S_{19} NP → · Adj NP [2,2] predict S_{17}

S_{20} NP → · PRP [2,2] predict S_{17}

S_{21} NP → · N [2,2] predict S_{17}

S_{22} V → · baking [2,2] predict S_{18}

S_{23} V → · are [2,2] predict S_{18}

S_{24} Adj → · baking [2,2] predict S_{19}

S_{25} PRP → · they [2,2] predict S_{20}

S_{26} N → · potatoes [2,2] predict S_{21}

Chart 3

S_{27} V → baking · [2,3] scan S_{22}

S_{28} Adj → baking [2,3] scan S_{24}

S_{29} VP → Aux V · NP [1,3] complete S_{18} w/ S_{27}

S_{30} NP → Adj · NP [2,3] complete S_{19} w/ S_{28}

S_{31} NP → · Adj NP [3,3] predict S_{29}, S_{30}

S_{32} NP → · PREP [3,3] predict S_{29}, S_{30}

Chart 3 cont.

S_{33} NP → · N [3,3] predict S_{29}, S_{30}

S_{34} Adj → · baking [3,3] predict S_{31}

S_{35} Prep → · thy [3,3] predict S_{32}

S_{36} N → · potatoes [3,3] predict S_{33}

Chart 4

S_{37} N → potatoes · [3,4] scan S_{36}

S_{38} NP → N · [3,4] complete S_{33} w/ S_{37}

S_{39} VP → Aux V NP · [1,4] complete S_{29} w/ S_{38}

S_{40} NP → Adj NP · [2,4] complete S_{30} w/ S_{38}

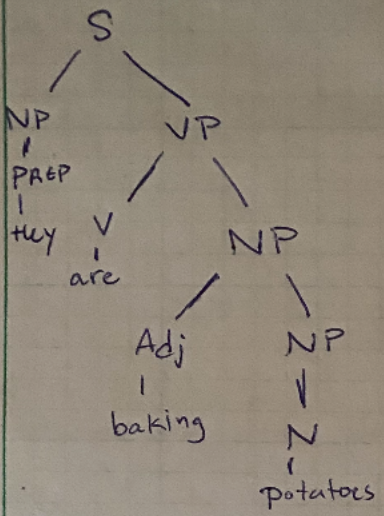
S_{41} S → NP VP · [0,4] complete S_{9} w/ S_{39}

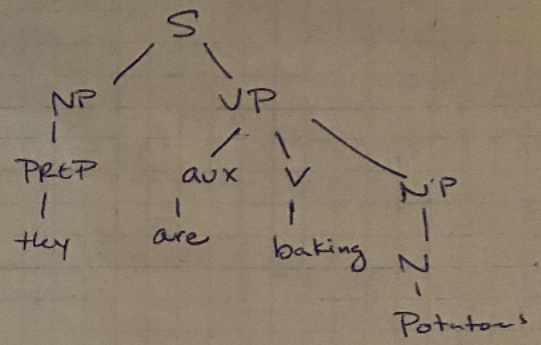
S_{42} VP → V NP · [1,4] complete S_{17} w/ S_{40}

S_{43} S → NP VP · [0,4] complete S_{9} w/ S_{42}

b.

Q2 Part b.


S
NP
PREP
they
VP
V
are
NP
Adj
baking
NP
N
potatoes


S
NP
PREP
they
VP
aux
are
V
baking
NP
N
Potatoes

See attached for parse trees.

Pr (left tree):

S → NP VP [1.0]
NP → PRP [0.1]
PRP → they [1]

VP → V NP [0.8]
V → are [0.5]
NP → ADJ NP [.3]
ADJ → baking [1.0]
NP→ N [0.6]
N → potatoes [1]

Pr(left tree) = 1*0.1*1*0.8*0.5*0.3*0.6*1 = 0.0072

Pr(right tree):

S → NP VP [1.0]
NP → PRP [0.1]
PRP → they [1]
VP → AUX V NP [0.2]
AUX → are [1.0]
V → baking [0.5]
NP → N [0.6]
N → potatoes [1]

Pr(right tree) = 1*0.1*1*0.2*1*0.5*0.6*1.0 = 0.006

Problem 3

a.

S → NP VP

NP → ADJ NP
VP → V NP
VP → TEMP1 NP
TEMP1 → AUX V

NP → they
NP → potatoes
Adj → baking
V → baking
V → are
Aux → are

1. General rule for form A → B, we can replace all of the rules with B in LHS with A directly So NP → N, N→ they, we remove N completely and add rule NP → they
2. For three or more non-terminals on RHS, we add a non-terminal LHS rule which splits the RHS into groups of two. So for VP → Aux V NP, we can create a temp rule which is T→ AUX V, and then rewrite the original VP as VP → T NP. We are essentially nesting this rule into two separate. If there are even more non-terminals, continue this process until each has two on RHS.

b.

	They	Are	Baking	Potatoes
	NP			S
		V, Aux	Temp1	VP
			Adj, V	NP, VP
				NP

TODO INSERT PHOTO OF TREE

Problem 4 – Transition Based Dependency Parsing

Stack	Buffer	A	Operation
Root	He, sent, her, a, funny, meme, today	{}	Shift

He, Root	Sent, her, a, funny, meme, today	{}	Left arc n_subj
Root	Sent, her, a, funny, meme, today	{(sent, nsubj, he)}	Shift
Sent Root	Her, a, funny, meme, today	{(sent, nsubj, he)}	R_arc iobj
Sent Root	A, funny, meme, today	{(sent, nsubj, he) (sent, iobj, her)}	Shift
A Sent Root	Funny, meme, today	{(sent, nsubj, he) (sent, iobj, her)}	Shift
Funny A Sent Root	Meme, today	{(sent, nsubj,he) (sent, iobj, her)}	L_arc amod
A Sent Root	Meme, today	{(sent, nsubj, he) (sent, iobj, her) (meme,amod,funny)}	L_arc det
Sent Root	Meme, today	{(sent, nsubj, he) (sent, iobj, her) (meme,amod,funny) (meme, det, a}	R_arc dobj
Sent	Today	(sent, nsubj, he) (sent, iobj, her) (meme,amod,funny) (meme, det, a) (sent, dobj, meme)}	R_arc advmod
Root	Sent	(sent, nsubj, he) (sent, iobj, her) (meme,amod,funny) (meme, det, a) (sent, dobj, meme) (sent, advmod, today)	R_arc pred
[]	root	{(he, nsubj, sent) (her, iobj, sent) (meme,amod,funny) (meme, det, a) (sent, dobj, meme) (sent, advmod, today) (sent, pred, root)}	shift
Root	[]	{(he, nsubj, sent) (her, iobj, sent) (meme,amod,funny) (meme, det, a) (sent, dobj, meme) (sent, advmod, today) (sent, pred, root)}	