Devin Liu

HW #6

1.

3b. TD

0	--	--	(S, the rat the cat chased fled)
1	Predict	S → NP VP	(NP VP, the rat the cat chased fled)
2	Predict	NP → D N ORC	(D N ORC VP, the rat the cat chased fled)
3	Match	D → the	(N ORC VP, rat the cat chased fled)
4	Match	N → rat	(ORC VP, the cat chased fled)
5	Predict	ORC → NP V	(NP V VP, the cat chased fled)
6	Predict	NP → D N	(D N V VP, the cat chased fled)
7	Match	D → the	(N V VP, cat chased fled)
8	Match	N → cat	(V VP, chased fled)
9	Match	V → chased	(VP, fled)
10	Predict	VP → V	(V, fled)
11	Match	V → fled	(ε, ε)

The maximum stack space here is **4** featured in steps 2 and 6.

3c. TD

0	--	--	(S, the rat the cat the dog bit chased fled)
1	Predict	S → NP VP	(NP VP, the rat the cat the dog bit chased fled)
2	Predict	NP → D N ORC	(D N ORC VP, the rat the cat the dog bit chased fled)
3	Match	D → the	(N ORC VP, rat the cat the dog bit chased fled)
4	Match	N → rat	(ORC VP, the cat the dog bit chased fled)
5	Predict	ORC → NP V	(NP V VP, the cat the dog bit chased fled)
6	Predict	NP → D N ORC	(D N ORC V VP, the cat the dog bit chased fled)
		.	
		.	
		.	
9	Predict	ORC → NP V	(NP V V VP, the dog bit chased fled)
10	Predict	NP → D N	(D N V V VP, the dog bit chased fled)
		.	
		.	
		.	
15	Predict	VP → V	(V, fled)
16	Match	V → fled	(ε, ε)

The maximum stack space here is **5** featured in steps 6 and 10.
Therefore, there is an increase in stack spaces of top-down parsers when parsing center-embedded structures.

Note: From here on out, ~ means a bar over the top of the character. Example: ~S = S bar

1b. LC

0	--	--	(~S, John's brother's dog barked)
1	Shift	NP → John	(NP ~S, 's brother's dog barked)
2	LCP	NP → NP Poss N	(~Poss ~N NP ~S, 's brother's dog barked)
3	Match	Poss → 's	(~N NP ~S, brother's dog barked)
4	Match	N → brother	(NP ~S, 's dog barked)
5	LCP	NP → NP Poss N	(~Poss ~N NP ~S, 's dog barked)
6	Match	Poss → 's	(~N NP ~S, dog barked)
7	Match	N → dog	(NP ~S, barked)
8	LCC	S → NP VP	(~VP, barked)
9	Shift	V → barked	(V ~VP, ε)
10	LCC	VP → V	(ε, ε)

The maximum stack space here is **4** featured in steps 2 and 5.

1c. LC

0	--	--	(~S, John's brother's wife's dog barked)
1	Shift	NP → John	(NP ~S, 's brother's wife's dog barked)
2	LCP	NP → NP Poss N	(~Poss ~N NP ~S, 's brother's wife's dog barked)
		.	
		.	
		.	
5	LCP	NP → NP Poss N	(~Poss ~N NP ~S, 's wife's dog barked)
		.	
		.	
		.	
8	LCP	NP → NP Poss N	(~Poss ~N NP ~S, 's dog barked)
		.	
		.	
		.	
13	LCC	VP → V	(ε, ε)

The maximum stack space here is **4** featured in steps 2, 5 and 8.
Therefore, there is no increase in stack spaces of left-corner parsers when parsing left-branching structures.

2b. LC

0	--	--	(~S, Mary chased the cat that bit the rat)
1	Shift	NP → Mary	(NP ~S, chased the cat that bit the rat)
2	LCC	S → NP VP	(~VP, chased the cat that bit the rat)
3	Shift	V → chased	(V ~VP, the cat that bit the rat)
4	LCC	VP → V NP	(~NP, the cat that bit the rat)
5	Shift	D → the	(D ~NP, cat that bit the rat)
6	LCC	NP → D N SRC	(~N ~SRC, cat that bit the rat)
7	Match	N → cat	(~SRC, that bit the rat)
8	Shift	THAT → that	(THAT ~SRC, bit the rat)
9	LCC	SRC → THAT VP	(~VP, bit the rat)
10	Shift	V → bit	(V ~VP, the rat)
11	LCC	VP → V NP	(~NP, the rat)
12	Shift	D → the	(D ~NP, rat)
13	LCC	NP → D N	(~N, rat)
14	Match	N → rat	(ε, ε)

The maximum stack space here is **2** featured in various steps.

2c. LC

0	--	--	(~S, Mary chased the cat that bit the rat that ate the cheese)
1	Shift	NP → Mary	(NP ~S, chased the cat that bit the rat that ate the cheese)
2	LCC	S → NP VP	(~VP, chased the cat that bit the rat that ate the cheese)
3	Shift	V → chased	(V ~VP, the cat that bit the rat that ate the cheese)
4	LCC	VP → V NP	(~NP, the cat that bit the rat that ate the cheese)
5	Shift	D → the	(D ~NP, cat that bit the rat that ate the cheese)
6	LCC	NP → D N SRC	(~N ~SRC, cat that bit the rat that ate the cheese)
7	Match	N → cat	(~SRC, that bit the rat that ate the cheese)
8	Shift	THAT → that	(THAT ~SRC, bit the rat that ate the cheese)
9	LCC	SRC → THAT VP	(~VP, bit the rat that ate the cheese)
10	Shift	V → bit	(V ~VP, the rat that ate the cheese)
11	LCC	VP → V NP	(~NP, the rat that ate the cheese)
12	Shift	D → the	(D ~NP, rat that ate the cheese)
13	LCC	NP → D N SRC	(~N ~SRC, rat that ate the cheese)
14	Match	N → rat	(~SRC, that ate the cheese)
15	Shift	THAT → that	(THAT ~SRC, ate the cheese)

16	LCC	SRC → THAT VP	(~VP, ate the cheese)
17	Shift	V → ate	(V ~VP, the cheese)
18	LCC	VP → V NP	(~NP, the cheese)
19	Shift	D → the	(D ~NP, cheese)
20	LCC	NP → D N	(~N, cheese)
21	Match	N → cheese	(ε, ε)

The maximum stack space here is **2** featured in various steps.
Therefore, there is no increase in stack spaces of left-corner parsers when parsing right-branching structures.

3b. LC

0	--	--	(~S, the rat the cat chased fled)
1	Shift	D → the	(D ~S, rat the cat chased fled)
2	LCP	NP → D N ORC	(~N ~ORC NP ~S, rat the cat chased fled)
3	Match	N → rat	(~ORC NP ~S, the cat chased fled)
4	Shift	D → the	(D ~ORC NP ~S, cat chased fled)
5	LCP	NP → D N	(~N NP ~ORC NP ~S, cat chased fled)
6	Match	N → cat	(NP ~ORC NP ~S, chased fled)
7	LCC	ORC → NP V	(~V NP ~S, chased fled)
8	Match	V → chased	(NP ~S, fled)
9	LCC	S → NP VP	(~VP, fled)
10	Shift	V → fled	(V ~VP, fled)
11	LCC	VP → V	(ε, ε)

The maximum stack space here is **5** featured in step 5.

3c. LC

0	--	--	(~S, the rat the cat the dog bit chased fled)
1	Shift	D → the	(D ~S, rat the cat the dog bit chased fled)
2	LCP	NP → D N ORC	(~N ~ORC NP ~S, rat the cat the dog bit chased fled)
3	Match	N → rat	(~ORC NP ~S, the cat the dog bit chased fled)
4	Shift	D → the	(D ~ORC NP ~S, cat the dog bit chased fled)
5	LCP	NP → D N ORC	(~N ~ORC NP ~ORC NP ~S, cat the dog bit chased fled)
6	Match	N → cat	(~ORC NP ~ORC NP ~S, the dog bit chased fled)
7	Shift	D → the	(D ~ORC NP ~ORC NP ~S, dog bit chased fled)

8	LCP	NP → D N	(~N NP ~ORC NP ~ORC NP ~S, dog bit chased fled)
9	Match	N → dog	(NP ~ORC NP ~ORC NP ~S, bit chased fled)
10	LCC	ORC → NP V	(~V NP ~ORC NP ~S, bit chased fled)
11	Match	V → bit	(NP ~ORC NP ~S, chased fled)
12	LCC	ORC → NP V	(~V NP ~S, chased fled)
13	Match	V → chased	(NP ~S, fled)
14	LCC	S → NP VP	(~VP, fled)
15	Shift	V → fled	(V ~VP, fled)
16	LCC	VP → V	(ε, ε)

The maximum stack space here is **7** featured in step 8.
Therefore, there is an increase in stack spaces of left-corner parsers when parsing center-embedding structures.

To summarize these configurations, here's the completed table:

	left-branching	right-branching	center-embedding
Human difficulty	No increase	No increase	Increase
Bottom-up parser	No increase	Increase	Increase
Top-down parser	Increase	No increase	**Increase**
Left-corner parser	**No increase**	**No increase**	**Increase**

2.

We know that the Martians will give a "yuck" reaction when the stack space of the bottom-up parser exceeds their memory limitation. The Martians give a "yuck" response to *Mary chased the cat that bit the rat* which has a maximum stack space of 8 when parsed bottom-up (we did this parsing in class so I will use this as an already proven fact). Using this information, we can safely conclude that sentences that need 8 or more stack spaces for bottom-up parsing should evoke a "yuck" reaction from the Martians. We will use this limitation to find the correct grammar.

6b. Hypothesis #1

0	--	--	(ε, Mary said quietly John ate)
1	Shift	NP → Mary	(NP, said quietly John ate)

2	Shift	SAID → said	(NP SAID, quietly John ate)
3	Shift	ADV → quietly	(NP SAID ADV, John ate)
4	Shift	NP → John	(NP SAID ADV NP, ate)
5	Shift	V → ate	(NP SAID ADV NP V, ε)
6	Reduce	VP → V	(NP SAID ADV NP VP, ε)
7	Reduce	S → NP VP	(NP SAID ADV S, ε)
8	Reduce	VP → SAID ADV S	(NP VP, ε)
9	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #1's grammar is **5** in steps 5 and 6.

6c. Hypothesis #1

0	--	--	(ε, John said slowly Mary said quietly John ate)
1	Shift	NP → John	(NP, said slowly Mary said quietly John ate)
		.	
		.	
		.	
4	Shift	NP → Mary	(NP SAID ADV NP, said quietly John ate)
		.	
		.	
		.	
7	Shift	NP → John	(NP SAID ADV NP SAID ADV NP, ate)
8	Shift	V → ate	(NP SAID ADV NP SAID ADV NP V, ε)
9	Reduce	VP → V	(NP SAID ADV NP SAID ADV NP VP, ε)
10	Reduce	S → NP VP	(NP SAID ADV NP SAID ADV S, ε)
11	Reduce	VP → SAID ADV S	(NP SAID ADV NP VP, ε)
12	Reduce	S → NP VP	(NP SAID ADV S, ε)
13	Reduce	VP → SAID ADV S	(NP VP, ε)
14	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #1's grammar is **8** in steps 8 and 9.

6d. Hypothesis #1

0	--	--	(ε, John said slowly John said loudly Mary said quietly John ate)
1	Shift	NP → John	(NP, said slowly John said loudly Mary said quietly John ate)
		.	
		.	
		.	
4	Shift	NP → John	(NP SAID ADV NP, said loudly Mary said quietly John ate)

		.	
		.	
		.	
7	Shift	NP → Mary	(NP SAID ADV NP SAID ADV NP, said quietly John ate)
		.	
		.	
		.	
10	Shift	NP → John	(NP SAID ADV NP SAID ADV NP SAID ADV NP, ate)
11	Shift	V → ate	(NP SAID ADV NP SAID ADV NP SAID ADV NP V, ε)
12	Reduce	VP → V	(NP SAID ADV NP SAID ADV NP SAID ADV NP VP, ε)
		.	
		.	
		.	
19	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #1's grammar is **11** in steps 11 and 12.

6b. Hypothesis #2

0	--	--	(ε, Mary said quietly John ate)
1	Shift	NP → Mary	(NP, said quietly John ate)
2	Shift	SAID → said	(NP SAID, quietly John ate)
3	Shift	ADV → quietly	(NP SAID ADV, John ate)
4	Reduce	X → SAID ADV	(NP X, John ate)
5	Shift	NP → John	(NP X NP, ate)
6	Shift	V → ate	(NP X NP V, ε)
7	Reduce	VP → V	(NP X NP VP, ε)
8	Reduce	S → NP VP	(NP X S, ε)
9	Reduce	VP → X S	(NP VP, ε)
10	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #2's grammar is **4** in steps 6 and 7.

6c. Hypothesis #2

0	--	--	(ε, John said slowly Mary said quietly John ate)
1	Shift	NP → John	(NP, said slowly Mary said quietly John ate)
		.	
		.	
		.	

4	Reduce	X → SAID ADV	(NP X, Mary said quietly John ate)
5	Shift	NP → Mary	(NP X NP, said quietly John ate)
		.	
		.	
		.	
9	Shift	NP → John	(NP X NP X NP, ate)
10	Shift	V → ate	(NP X NP X NP V, ε)
		.	
		.	
		.	
16	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #2's grammar is **6** in step 10.

6d. Hypothesis #2

0	--	--	(ε, John said slowly John said loudly Mary said quietly John ate)
1	Shift	NP → John	(NP, said slowly John said loudly Mary said quietly John ate)
		.	
		.	
		.	
5	Shift	NP → John	(NP X NP, said loudly Mary said quietly John ate)
		.	
		.	
		.	
9	Shift	NP → Mary	(NP X NP X NP, said quietly John ate)
		.	
		.	
		.	
13	Shift	NP → John	(NP X NP X NP X NP, ate)
14	Shift	V → ate	(NP X NP X NP X NP V, ate)
15	Reduce	VP → V	(NP X NP X NP X NP VP, ε)
		.	
		.	
		.	
22	Reduce	S → NP VP	(S, ε)

The maximum stack space using Hypothesis #2's grammar is **8** in steps 14 and 15.

After parsing sentences 6b, 6c and 6d using the bottom-up method and two different grammars, here are the findings represented in a table:

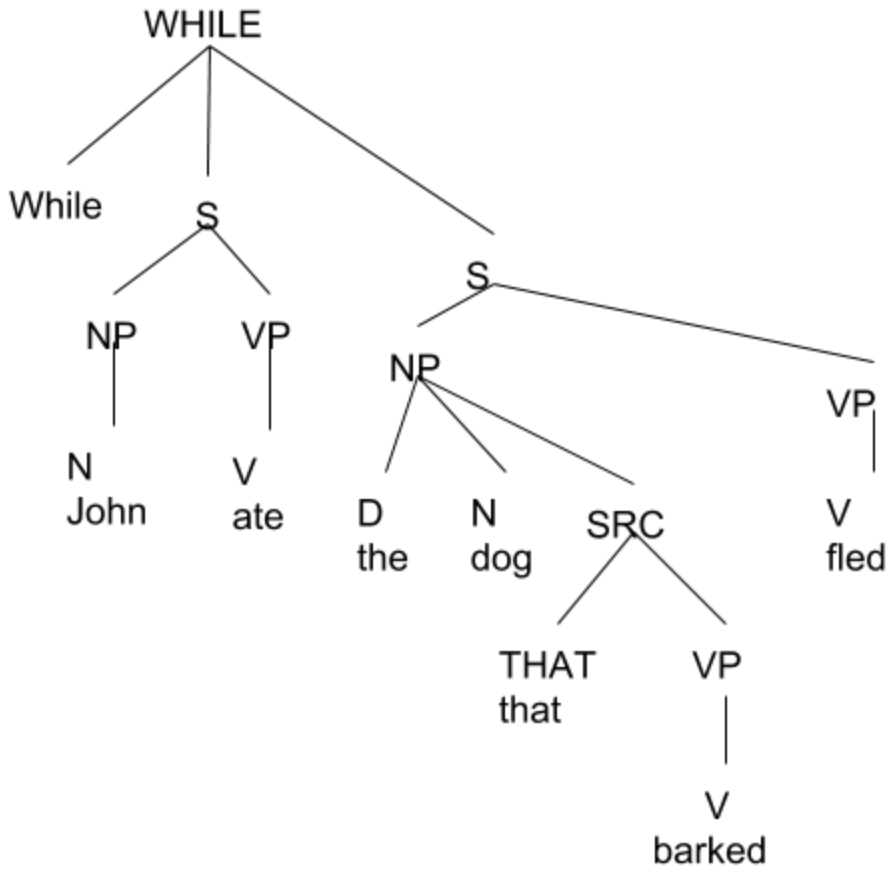
	6b	6c	6d
Hypothesis #1	5 stack spaces	8	11
Hypothesis #2	4	6	8

Recall that any sentences which require a stack space of 8 or more will be considered “yuck” by a Martian. If we assume that Hypothesis #1 is correct, then both 6c and 6d should be considered “yuck”. However in reality, only 6d is “yuck”. This is a contradiction. In contrast using Hypothesis #2, only 6d is “yuck”, which corroborates with reality. **Therefore, Hypothesis #2 is correct.**

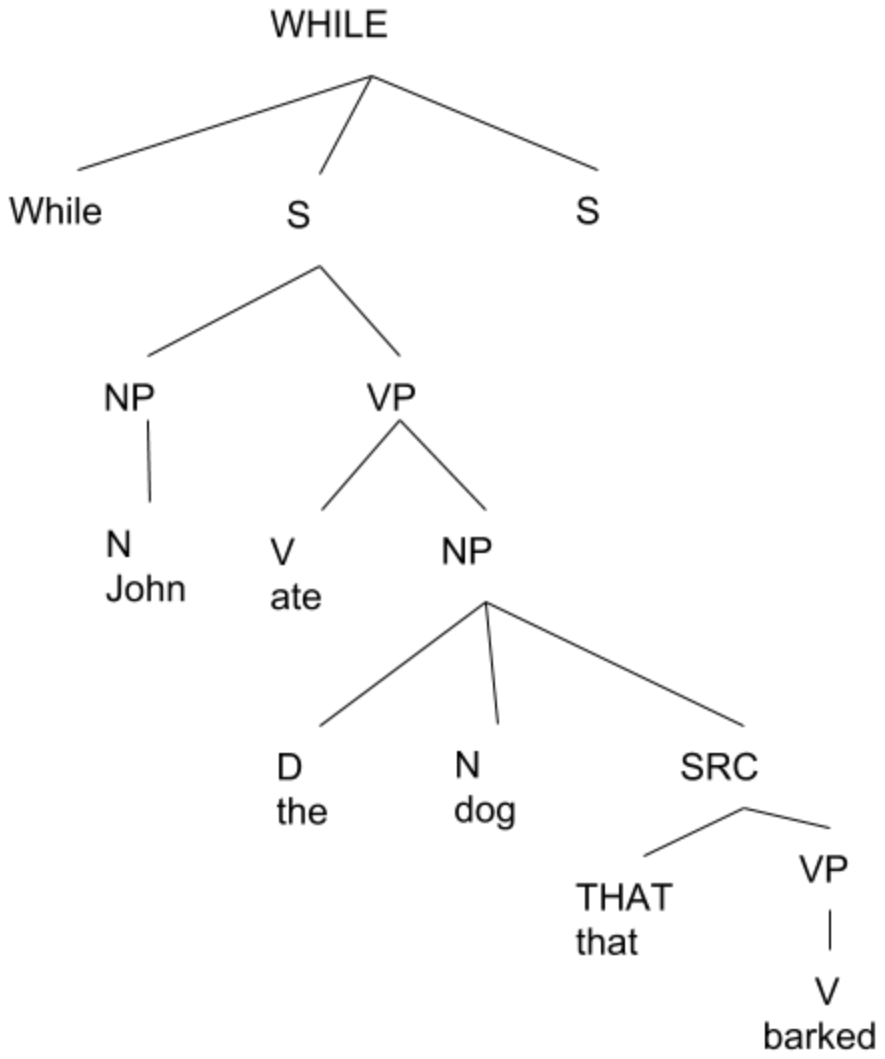
3.

A.

Correct tree:



Partial tree:



B.

0	--	--	(ε, While John ate the dog that barked fled)
1	Shift	WHILE → while	(WHILE, John ate the dog that barked fled)
2	Shift	NP → John	(WHILE NP, ate the dog that barked fled)
3	Shift	V → ate	(WHILE NP V, the dog that barked fled)
4	Reduce	VP → V	(WHILE NP VP, the dog that barked fled)
5	Reduce	S → NP VP	(WHILE S, the dog that barked fled)
6	Shift	D → the	(WHILE S D, dog that barked fled)
7	Shift	N → dog	(WHILE S D N, that barked fled)
8	Shift	THAT → that	(WHILE S D N THAT, barked fled)
9	Shift	V → barked	(WHILE S D N THAT V, fled)
10	Reduce	VP → V	(WHILE S D N THAT VP, fled)

11	Reduce	SRC → THAT VP	(WHILE S D N SRC, fled)
12	Reduce	NP → D N SRC	(WHILE S NP, fled)
13	Shift	V → fled	(WHILE S NP V, ε)
14	Reduce	VP → V	(WHILE S NP VP, ε)
15	Reduce	S → NP VP	(WHILE S S, ε)
16	Reduce	S → WHILE S S	(S, ε)

The first step where someone could head off in the wrong direction in this parsing is after step 3. The incorrect action here would be to shift to the next word as opposed to reducing. Here are some of the steps of the wrong configuration:

3	Shift	V → ate	(WHILE NP V, the dog that barked fled)
4	Shift	D → the	(WHILE NP V D, dog that barked fled)
5	Shift	N → dog	(WHILE NP V D N, that barked fled)
6	Shift	THAT → that	(WHILE NP V D N THAT, barked fled)
7	Shift	V → barked	(WHILE NP V D N THAT V, fled)
8	Reduce	VP → V	(WHILE NP V D N THAT VP, fled)
9	Reduce	SRC → THAT VP	(WHILE NP V D N SRC, fled)
10	Reduce	NP → D N SRC	(WHILE NP V NP, fled)
11	Reduce	VP → V NP	(WHILE NP VP, fled)
12	Reduce	S → NP VP	(WHILE S, fled)

After step 12 of this configuration, a dead end is reached.

C.

0	--	--	(S, While John ate the dog that barked fled)
1	Predict	S → WHILE S S	(WHILE S S, While John ate the dog that barked fled)
2	Match	WHILE → while	(S S, John ate the dog that barked fled)
3	Predict	S → NP VP	(NP VP S, John ate the dog that barked fled)
4	Match	NP → John	(VP S, ate the dog that barked fled)
5	Predict	VP → V	(V S, ate the dog that barked fled)
6	Match	V → ate	(S, the dog that barked fled)
7	Predict	S → NP VP	(NP VP, the dog that barked fled)
8	Predict	NP → D N SRC	(D N SRC VP, the dog that barked fled)
9	Match	D → the	(N SRC VP, dog that barked fled)
10	Match	N → dog	(SRC VP, that barked fled)
11	Predict	SRC → THAT VP	(THAT VP VP, that barked fled)
12	Match	THAT → that	(VP VP, barked fled)
13	Predict	VP → V	(V VP, barked fled)
14	Match	V → barked	(VP, fled)
15	Predict	VP → V	(V, fled)

16	Match	V → fled	(ε, ε)

The first step where someone could head off in the wrong direction in this parsing is step 5. The incorrect action here would be to predict *VP → V NP* as opposed to predicting *VP → V*. Here are some of the steps of the wrong configuration:

5	Predict	VP → V NP	(V NP S, ate the dog that barked fled)
6	Match	V → ate	(NP S, the dog that barked fled)
7	Predict	NP → D N SRC	(D N SRC S, the dog that barked fled)
8	Match	D → the	(N SRC S, dog that barked fled)
9	Match	N → dog	(SRC S, that barked fled)
10	Predict	SRC → THAT VP	(THAT VP S, that barked fled)
11	Match	THAT → that	(VP S, barked fled)
12	Predict	VP → V	(V S, barked fled)
13	Match	V → barked	(S, fled)

After step 13, a dead end is reached.

D.

We could use the direction of the backtracking to figure out what kind of parser humans are. If the individual's eyes are backtracking from right to left, then they are a top-down parser because they are backtracking towards the start of the sentence, which is equivalent to backtracking upwards in a tree. If the individual's eyes are backtracking from left to right, then they are implementing bottom-up parser because they are backtracking towards the end of the sentence, which is equivalent to the bottom of the tree.

4.

A.

0	--	--	(0A, aaaacbbbb)
1	Consume	0A, a → 1A	(1A, aaacbbbb)
2	Consume	1A, a → 2A	(2A, aacbbbb)
3	Consume	2A, a → 3A	(3A, acbbbb)
4	Consume	3A, a → 4A	(4A, cbbbb)
5	Consume	4A, c → 4B	(4B, bbbb)
6	Consume	4B, b → 3B	(3B, bbb)
7	Consume	3B, b → 2B	(2B, bb)
8	Consume	2B, b → 1B	(1B, b)
9	Consume	1B, b → 0B	(0B, ε)

B.

0	--	--	(S, aaaacbbbb)

1	Predict	S → ASB	(ASB, aaaacbbbb)
2	Match	A → a	(SB, aaacbbbb)
3	Predict	S → ASB	(ASBB, aaacbbbb)
4	Match	A → a	(SBB, aacbbbb)
5	Predict	S → ASB	(ASBBB, aacbbbb)
6	Match	A → a	(SBBB, acbbbb)
7	Predict	S → ASB	(ASBBBB, acbbbb)
8	Match	A → a	(SBBBB, cbbbb)
9	Predict	S → C	(CBBBB, cbbbb)
10	Match	C → c	(BBBB, bbbb)
11	Match	B → b	(BBB, bbb)
12	Match	B → b	(BB, bb)
13	Match	B → b	(B, b)
14	Match	B → b	(ε, ε)

C.

0	--	--	(ε, aaaacbbbb)
1	Shift	A → a	(A, aaacbbbb)
2	Shift	A → a	(AA, aacbbbb)
3	Shift	A → a	(AAA, acbbbb)
4	Shift	A → a	(AAAA, cbbbb)
5	Shift	C → c	(AAAAC, bbbb)
6	Reduce	S → C	(AAAAS, bbbb)
7	Shift	B → b	(AAAASB, bbb)
8	Reduce	S → ASB	(AAAS, bbb)
9	Shift	B → b	(AAASB, bb)
10	Reduce	S → ASB	(AAS, bb)
11	Shift	B → b	(AASB, b)
12	Reduce	S → ASB	(AS, b)
13	Shift	B → b	(ASB, ε)
14	Reduce	S → ASB	(S, ε)

Note: ~S = S bar, ~A = A bar, etc… for part D

D.

0	--	--	(~S, aaaacbbbb)
1	Shift	A → a	(A ~S, aaacbbbb)
2	LCC	S → ASB	(~S ~B, aaacbbbb)
3	Shift	A → a	(A ~S ~B, aacbbbb)
4	LCC	S → ASB	(~S ~B ~B, aacbbbb)
5	Shift	A → a	(A ~S ~B ~B, acbbbb)
6	LCC	S → ASB	(~S ~B ~B ~B, acbbbb)
7	Shift	A → a	(A ~S ~B ~B ~B, cbbbb)
8	LCC	S → ASB	(~S ~B ~B ~B ~B, cbbbb)

9	Shift	C → c	(C ~S ~B ~B ~B ~B, bbbb)
10	LCC	S → C	(~B ~B ~B ~B, bbbb)
11	Match	B → b	(~B ~B ~B, bbb)
12	Match	B → b	(~B ~B, bb)
13	Match	B → b	(~B, b)
14	Match	B → b	(ε, ε)