SJSU Students CS 154 Section 01 1:30-2:45 04/06/11

Homework #3

1. Context free grammar for palindromes:

S --> aSa | bSb | cSc | a | b | c | ϵ

Size									
	RHS								
\rightarrow	aSa								
\rightarrow	bSb								
\rightarrow	cSc								
\rightarrow	a								
\rightarrow	b								
\rightarrow	c								
\rightarrow	ε								
	Size \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow \rightarrow								

First of all, to enter the grammar into JFLAP, select grammar from the main JFLAP menu. Then, add each rule into the boxes provided. The left most box takes the left side of the productions. Next, the middle box will always take the arrow for the production. Finally, the right box will take the right side of the production.



After entering the grammar, click convert from the topmost menu and choose convert to PDA. The above is picture is what is obtained after converting. In the first transition from q0 to q1, S, the string that needs to be processed by the automaton, and the variable Z are pushed onto the stack. The transition also says that ε is all that is required to get from q0 to q1. Now, notice that the only transitions on q1 are those that involve popping symbols off the stack. For example: in state q1 a, a; ε will pop 'a' off the stack if 'a' is the next symbol that needs to processed. The final transition between q1 and q2 can only be reached if it is the end of the string being processed. Also, this transition says that the transition can only be followed to q2 if the variable Z can be popped off the stack. So, in order to reach the accept state, the stack must contain only the variable Z, while the entirety of S must be popped off.



The next step involves adding the transitions in state q1 that push variables and terminals onto the stack. Having rules to replace every variable in S with terminals is necessary to get to the accept state, since every transition in q1 that pops a variable off the stack requires a terminal input. For example: a, a; ε will only work if there is variable 'a' in the input and a variable 'a' on the stack. To add the necessary transitions, click on one of the production rules in the table on the left and then click the 'Create Selected' button. This will add the transition for the production in q1. As shown in the picture, for the first production S \rightarrow aSa, the transition rule generated is: ε , S; aSa. This rule tells us that on an empty string input, pop S off the stack and push aSa onto the stack.



After repeating the process of adding production as transitions, the PDA pictured above is generated.

2. Context free grammar:

 $S \rightarrow P$ $P \rightarrow \langle p \rangle A \langle /p \rangle P \mid \varepsilon$ $A \rightarrow \langle i \rangle B \langle /i \rangle A \mid \langle b \rangle I \langle /b \rangle A \mid \varepsilon$ $B \rightarrow \langle b \rangle \langle /b \rangle B \mid \varepsilon$ $I \rightarrow \langle i \rangle \langle /i \rangle I \mid \varepsilon$

Editor Lambda Removal									
Table Text	Size								
		\frown							
LHS		RHS		Do Ster	Do	All Procee	d Export		
S	\rightarrow	Р		Select va	riable	s that derive I	ambda.		
Р	\rightarrow	AP		Click pro	ductio	ns; the LHS va	ariable will be added.		
Р	\rightarrow	ε	Sector Sector	Delete	Comp	lete Selected			
А	\rightarrow	<i>B</i> A		LHS			RHS		
А	\rightarrow	I A							
А	\rightarrow	ε							
В	\rightarrow	 B							
В	\rightarrow	ε							
Ι	\rightarrow	<i><i>i>I</i></i>	and the second						
I	\rightarrow	ε							

Once again, the first step is to enter the grammar into JFLAP. After that, click 'Transform Grammar' from the Convert menu at the top. This will cause the above image to appear.

Editor	ambd	la Removal						
Table Text	Size							
		\frown						
LHS		RHS	Do Step	Do A		Proceed	Export	
S	\rightarrow	Р	Select va	riable	s th	at derive la	mbda.	
Р	\rightarrow	AP	l added! 4 Set that d	4 more erives	e va s lar	riable(s) ne nbda: [l]	eded.	
Р	\rightarrow	3	Delete	Com	olete	e Selected		
А	\rightarrow	<i>B</i> A	LHS				RHS	
А	\rightarrow	I A						
А	\rightarrow	ε						
В	\rightarrow	 B						
В	\rightarrow	ε						
I	\rightarrow	<i></i> I						
I	\rightarrow	ε						

Now, variables that derive ε must be selected. To select a variable that derives ε , just click on the production rule, where that variable is on the left side of the arrow.

Editor	Lambd	la Removal									
Table Tex	xt Size-		_								
LHS		RH	8		Do Step	Do /	AII	Proceed	Export		
S	\rightarrow	Р		N	/lodify the	e gran	ıma	r to remov	e lambda:	s.	
Р	\rightarrow	A <td>o>P</td> <td>4</td> <td>l more re Set that d</td> <td>move erives</td> <td>(s), a Han</td> <td>and 11 mo 1bda: [A, B</td> <td>re additioı , I, P, S]</td> <td>n(s) needed.</td> <td></td>	o>P	4	l more re Set that d	move erives	(s), a Han	and 11 mo 1bda: [A, B	re additioı , I, P, S]	n(s) needed.	
Р	\rightarrow	ε			Delete	Comp	lete	Selected			
А	\rightarrow	<i>B</i>	>A		LHS				RHS		Ι.
А	\rightarrow	I	>A	S		\rightarrow	Ρ				-
А	\rightarrow	ε		P		\rightarrow	<p< td=""><td>>A</td><td>>P</td><td></td><td></td></p<>	>A	>P		
В	\rightarrow		B	P		\rightarrow	ε				
В	\rightarrow	ε		A		\rightarrow	<i< td=""><td>>B</td><td>A</td><td></td><td></td></i<>	>B	A		
Ι	\rightarrow	<i><i>i>I</i></i>		A		\rightarrow	<b< td=""><td>>I</td><td>A</td><td></td><td>=</td></b<>	>I	A		=
Ι	\rightarrow	ε		A		\rightarrow	ε				
				В		\rightarrow	<b< td=""><td>>>F</td><td>3</td><td></td><td></td></b<>	>>F	3		
				В		\rightarrow	ε				
				Ι		\rightarrow	<i< td=""><td>>>/i>I</td><td></td><td></td><td></td></i<>	>>/i>I			
				Ι		\rightarrow	3				-

After all the variables that derive $\boldsymbol{\epsilon}$ have been selected, the above image appears.

LHS		RHS	Do Step	Do	All Proceed Export									
S	\rightarrow	Р	Modify th	e gran	nmar to remove lambdas.									
Р	\rightarrow	AP	0 more remove(s), and 11 more addition(s) needed. Set that derives lambda: [A, B, I, P, S]											
Р	\rightarrow	ε	Delete	Comp	plete Selected									
А	\rightarrow	<i>B</i> A	LHS		RHS									
А	\rightarrow	I A	S	\rightarrow	Р									
А	\rightarrow	ε	Р	\rightarrow	AP									
В	\rightarrow	 B	A	\rightarrow	<i>B</i> A									
В	\rightarrow	ε	A	\rightarrow	I A									
I	\rightarrow	<i>/i>I</i>	В	\rightarrow	 B									
I	\rightarrow	ε	[\rightarrow	<i><i>/i>I</i></i>									

Next, delete all the rules that lead to ε in the table of productions that was generated on the right. This can be done by selecting a rule and clicking the delete button.

Editor	ambo	la Removal				
Table Text	Size					
		\frown				
LHS		RHS	l	Do Step	Do	All Proceed Export
S	\rightarrow	Р	A CONTRACTOR	Modify th	e gran	nmar to remove lambdas.
Р	\rightarrow	AP		0 more re Set that o	emove lerives	(s), and 10 more addition(s) needed. ; lambda: [A, B, I, P, S]
Р	\rightarrow	ε		Delete	Comp	lete Selected
А	\rightarrow	<i>B</i> A	A CONTRACT	LHS		RHS
А	\rightarrow	I A	- Andrews	S	\rightarrow	Р
А	\rightarrow	ε		Р	\rightarrow	AP
В	\rightarrow	 B	- Andrews	А	\rightarrow	<i>B</i> A
В	\rightarrow	ε		А	\rightarrow	I A
I	\rightarrow	<i></i> I	A CONTRACTOR	В	\rightarrow	 B
I	\rightarrow	ε	A STATE OF A	Ι	\rightarrow	<i><i>/i>I</i></i>
				I	\rightarrow	<i></i>

LHS		RHS		Do Step	Do /		Proceed	Export	
S	\rightarrow	Р		Lambda i	remov	/al co	omplete.		
Р	\rightarrow	AP		"Proceed Set that d	l" or " Ierive	'Expo s Ian	ort" availat nbda: [A, B	ole. , I, P, S]	
Р	\rightarrow	ε		Delete	Com	plete	Selected		
А	\rightarrow	<i>B</i> A		LHS				RHS	
А	\rightarrow	I A	S		\rightarrow	Ρ			
А	\rightarrow	ε	P		\rightarrow	<p?< td=""><td>>A</td><td>P</td><td></td></p?<>	>A	P	
В	\rightarrow	 B	А		\rightarrow	<i></i>	>BA	ł	
В	\rightarrow	ε	А		\rightarrow	<b?< td=""><td>>I</td><td>4</td><td></td></b?<>	>I	4	
I	\rightarrow	<i>∕i>I</i>	В		\rightarrow	<b< td=""><td>>>/b>B</td><td></td><td></td></b<>	>>/b>B		
T	\rightarrow	ε	I		$ \rightarrow$	<i></i>	>I		
			Ι		\rightarrow	<i></i>	>		
			В		\rightarrow	<b< td=""><td>></td><td></td><td></td></b<>	>		
			A		\rightarrow	<b< td=""><td>></td><td></td><td>=</td></b<>	>		=
			A		\rightarrow	<b< td=""><td>>>/b>A</td><td></td><td></td></b<>	>>/b>A		
			A		\rightarrow	<b< td=""><td>>I</td><td></td><td></td></b<>	>I		
			A		\rightarrow	<i></i>	>		
			A		\rightarrow	<i></i>	>A		
			A		\rightarrow	<i></i>	>B		
			Р		\rightarrow	<p?< td=""><td>></td><td></td><td></td></p?<>	>		
			Р		\rightarrow	<p?< td=""><td>>P</td><td></td><td></td></p?<>	>P		
			Р		\rightarrow	<p?< td=""><td>>A</td><td></td><td></td></p?<>	>A		
			2			-	-		

After adding all of the necessary rules, the above image is produced. Now, all ϵ rules have been removed and the new grammar has been fully modified to accept the same set of strings as the original language. Now the next step is to remove unit rules.

LHS		RHS			Do Step	Do All	1	Proceed	Export		
S	\rightarrow	Р	-	C	omplete un	it produ	ict	ion visualiz	ation.		
Р	\rightarrow	AP		1	more trans	ition(s)	ne	eeded.			
А	\rightarrow	<i>B</i> A			N 27						
А	\rightarrow	I A							•		^
В	\rightarrow	 B					•)				
I	\rightarrow	<i><i>/i>I</i></i>									
I	\rightarrow	<i></i>	_							<u>~</u>	
В	\rightarrow										
А	\rightarrow						B)				
А	\rightarrow	 A							P		
А	\rightarrow	I			•						▼ ↓
А	\rightarrow	<i></i>		Γ	Automator	Size					
А	\rightarrow	<i><i>A</i></i>									
А	\rightarrow	<i>B</i>			Delete Co	mplete	Se	elected			
Р	\rightarrow			-	LHS					RHS	
р	\rightarrow	<n></n> P	•								

After clicking the Proceed button, the above image is displayed. The graph on the right has a node for every variable in the grammar.



Now for each unit rule add a transition arrow. So, for the unit rule $S \rightarrow P$, add a transition arrow from state S to state P as shown above.

Editor	Lambda Removal	Unit Removal		
LHS	RHS		Do Step Do	o All Proceed Export
S	\rightarrow P	Mo	odify the gram	nmar to remove unit productions.
Р	\rightarrow A <th>>P ^{0 n}</th> <th>nore removes</th> <th>s, and 4 more additions needed.</th>	>P ^{0 n}	nore removes	s, and 4 more additions needed.
А	$\rightarrow B$	>A	R 27	
А	$\rightarrow I$	>A		. • •
В	$\rightarrow <\!$	B		•
Ι	\rightarrow <i><i>/i>I</i></i>			•
I	\rightarrow <i></i>			
В	\rightarrow 	·	Automaton Siz	ze
А	\rightarrow 			
А	\rightarrow 	A	Delete Comp	plete Selected
А	$\rightarrow I$	>P		$\rightarrow \langle p \rangle A \langle p \rangle P$
А	\rightarrow <i></i>	A	· –	$\rightarrow BA$
А	$\rightarrow /i>/i>/i>/i$	А А	· –	\rightarrow I A
А	$\rightarrow B$	>B	3 —	→ B
Р	$\rightarrow $	I		→ <i></i> I
D	$\rightarrow < n > /n >$.p 💌 🗖		▼

After that, delete any unit production rules in the bottom right table the same way

 ϵ rules were deleted. So, S \rightarrow P is deleted from the table.

Editor	ambda Removal 🛛 Unit F	emoval
LHS A	$\rightarrow <_{1>BA}}$	Do Step Do All Proceed Export
А	$\rightarrow <\!$	Unit removal complete. "Proceed" or "Export" available.
В	$\rightarrow <\!$	
Ι	$\rightarrow I$	-
Ι	$\rightarrow <$	
В	$\rightarrow <\!$	•
А	$\rightarrow <\!$	
А	$\rightarrow <\!$	Automaton Size
А	$\rightarrow <\!$	
А	$\rightarrow <$	Delete Complete Selected
А	$\rightarrow A$	LHS P P RHS
А	$\rightarrow B$	$P \longrightarrow A$
Р	$\rightarrow <\!$	$S \longrightarrow AP$
Р	$\longrightarrow <\!$	$S \longrightarrow $
Р	$\rightarrow A$	$S \longrightarrow /p>P$
		\checkmark S \rightarrow A

Now, just like in Epsilon removal, additional rules need to be added to the grammar to make it equal to the original grammar. So, since $S \rightarrow P$ was deleted, a new production with S on the left needs to be added for every production that had P on the left. This is why there are four new rules with S on the left added into the new grammar.

Editor	Lambo	la Removal	Unit Remova	al 🛛	Cho	msky Co	onvert	ter								
LHS		R	HS			Convert	Selec	ted	Do All	Wh	at's L	.eft?	1	Expor	t	
S	\rightarrow	A <td>o>P</td> <td>-</td> <td></td> <td colspan="8">Welcome to the Chomsky converter.</td> <td></td>	o>P	-		Welcome to the Chomsky converter.										
S	\rightarrow					20 production(s) must be converted.										
s	\rightarrow		P		s		\rightarrow		>A </td <td>p>P</td> <td>,</td> <td></td> <td></td> <td></td> <td></td> <td> -</td>	p>P	,					 -
S	\rightarrow	A <td>></td> <td></td> <td>s</td> <td></td> <td>\rightarrow</td> <td><p?< td=""><td>></td><td>></td><td></td><td></td><td></td><td></td><td></td><td></td></p?<></td>	>		s		\rightarrow	<p?< td=""><td>></td><td>></td><td></td><td></td><td></td><td></td><td></td><td></td></p?<>	>	>						
Р	\rightarrow	A <td>></td> <td></td> <td>s</td> <td></td> <td>\rightarrow</td> <td><p?< td=""><td>></td><td>>P</td><td></td><td></td><td></td><td></td><td></td><td></td></p?<></td>	>		s		\rightarrow	<p?< td=""><td>></td><td>>P</td><td></td><td></td><td></td><td></td><td></td><td></td></p?<>	>	>P						
Р	\rightarrow		P	_	S		\rightarrow	<p?< td=""><td>>A<!--</td--><td>p></td><td></td><td></td><td></td><td></td><td></td><td></td></td></p?<>	>A </td <td>p></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	p>						
Р	\rightarrow				Р		\rightarrow	<p?< td=""><td>>A<!--</td--><td>p></td><td></td><td></td><td></td><td></td><td></td><td>=</td></td></p?<>	>A </td <td>p></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td>=</td>	p>						=
А	\rightarrow	<i>B</i>	>		Р		\rightarrow	<p?< td=""><td>></td><td>>P</td><td></td><td></td><td></td><td></td><td></td><td></td></p?<>	>	>P						
А	\rightarrow	<i></i>	A		Р		\rightarrow	<p?< td=""><td>></td><td>></td><td></td><td></td><td></td><td></td><td></td><td></td></p?<>	>	>						
А	\rightarrow	<i></i>			Α		\rightarrow	<i></i>	-B <td>></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	>						
А	\rightarrow	I	>		Α		\rightarrow	<i></i>		A						-
А	\rightarrow		A		Α		\rightarrow	<i></i>								
А	\rightarrow				Α		\rightarrow	<b?< td=""><td>>I<td>></td><td></td><td></td><td></td><td></td><td></td><td></td></td></b?<>	>I <td>></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td>	>						
В	\rightarrow				Α		\rightarrow	<b?< td=""><td>></td><td>>A</td><td></td><td></td><td></td><td></td><td></td><td></td></b?<>	>	>A						
I	\rightarrow	<i></i>			Α		\rightarrow	<b?< td=""><td>></td><td>></td><td></td><td></td><td></td><td></td><td></td><td></td></b?<>	>	>						
I	\rightarrow	<i>/i>I</i>		Ŧ	Ъ		\ \	1	/1_>							-

After that, useless productions would be removed. However, since our grammar did not contain any useless production, JFLAP proceeds straight to the Chomsky Converter.

Editor	Lambda Removal 🛛 Unit Removal	Cho	msl	ky Converte	er					
LHS	RHS			Convert	Sele	cted [Do All	What's Left?	Export	
S	$\rightarrow AP$	_		Replacen	nent	produc	tion(s) highlighted.		
S	\rightarrow			20 produc	ction	i(s) mu	st be c	onverted.	9	
S	$\longrightarrow <\!$		s		\rightarrow	B(<)B(p)B(>)AB(<)B(/)H	B(p)B(>)P
S	$\rightarrow <\!\!p\!\!>\!\!A\!<\!\!/p\!\!>$		E	8(<)	\rightarrow	<				
Р	$\rightarrow <\!\!p\!\!>\!\!A<\!\!/p\!\!>$		E	8(p)	\rightarrow	р				
Р	$\rightarrow <\!\!p\!\!>\!\!<\!\!p\!\!>\!\!P$	_	E	3(>)	\rightarrow	>				
Р	$\rightarrow <\!\!p\!\!>\!\!<\!\!/p\!\!>$		E	3(/)	\rightarrow	/				
А	$\rightarrow B$		S		\rightarrow	-		>		
А	$\rightarrow i>A$		S		\rightarrow	-		P		
А	\rightarrow <i>></i>		S		\rightarrow		A 1</td <td>o></td> <td></td> <td></td>	o>		
А	$\rightarrow I$		P	1	\rightarrow		A 1</td <td>)></td> <td></td> <td></td>)>		
А	$\rightarrow <\!\!b\!\!>\!\!<\!\!b\!\!>\!\!A$		P		\rightarrow	-		P		
А	$\rightarrow <\!\!b\!\!>\!\!<\!\!b\!\!>$		P		\rightarrow	-		>		
В	$\rightarrow <\!\!b\!\!>\!\!<\!\!b\!\!>$		A	1	\rightarrow	- <i>I</i>	B	>		
I	$\rightarrow i>$		A	1	\rightarrow	<i></i>		4		
Т		-					· <i>P</i> ·			

Now every production rule must have the form: $A \rightarrow BC$ or $A \rightarrow a$. In these rules, A, B, and C are variables and 'a' is a terminal. One way to do this is to add more production rules manually and introduce new variables to get each production into CNF. The other way is to select a production and hit the 'Convert Selected' button. After converting every production that is not in CNF, the final grammar is obtained and the conversion to CNF is completed.

When the starting grammar for problem #2 is converted into CNF, JFLAP throws an error saying: 26 variables available, but 73 needed. So, the grammar entered into JFLAP needs be a little simplified to produce fewer variables in the CNF. Context free grammar entered into JFLAP:

 $S \rightarrow P$ $P \rightarrow CADP | \varepsilon$ $A \rightarrow FBGA | JIKA | \varepsilon$ $B \rightarrow JKB | \varepsilon$ $I \rightarrow FGI | \varepsilon$ $C \rightarrow p$ $D \rightarrow /p$ $F \rightarrow i$ $G \rightarrow /i$ $J \rightarrow b$ $K \rightarrow /b$

Note: in this grammar, the terminals < and > were excluded, so that JFLAP would not throw the error previously mentioned. So, those terminals are just assumed to be around all occurrences of the terminals p, /p, b, /b, i, and /i. The CNF generated from this grammar was used for problem #3.

2	
3	
-	1

Editor								
Table Text	Table Text Size							
LHS		RHS						
S	\rightarrow	CO						
S	\rightarrow	CH						
S	\rightarrow	CD						
S	\rightarrow	CN						
А	\rightarrow	FM						
м	\rightarrow	BU						
А	\rightarrow	Л						
L	\rightarrow	IQ						
В	\rightarrow	JW						
W	\rightarrow	KB						
А	\rightarrow	FV						
V	\rightarrow	BG						
А	\rightarrow	FG						

Once again, we start by inputting our grammar into JFLAP. This time it's the

CNF of the grammar from problem 2.

Editor	CYK	Parse			
Table Text Size					
			\sim		
Start	t Ste	p Nonin	werted Tree		
Input					
		вце			
<u>CH0</u>	\rightarrow	CO			
2	,	au			
S	\rightarrow	СН			
S	\rightarrow	CD			
S	\rightarrow	CN			
А	\rightarrow	FM			
м	\rightarrow	BU			
А	\rightarrow	Л			
L	\rightarrow	IO			

After that, go to the input menu and choose CYK Parse to get to screen shown in the above image.

Editor	C	YK Par	se					
Table	Table Text Size							
	tart	Step	No	ninverted Tree 🗸 🔻				
Input	pbi/i/ł	o/ppb/bi	i/ib/b	//ppib/b/i/p				
String	g is A	ccepte	d!					
LHS		RHS						
S	\rightarrow	СО						
S	\rightarrow	CH						
S	\rightarrow	CD						
S	\rightarrow	CN		S				
А	\rightarrow	FM						
М	\rightarrow	BU						
Λ		π	-					

The next step is to enter the string that will be CYK parsed. In our case, since the grammar was modified so that JFLAP would not throw an error, the input string that is provided in the homework problem needs to also be modified. So, the terminals < and > need to discarded from the string.

Homework input string provided:

<i><i><i><i><i><i><i><i><i><i>

Homework input string modified:

pbi/i/b/ppb/bi/ib/b/ppib/b/i/p

The modified version is the one we will be inputting into JFLAP.

The picture above is obtained after entering in the string and hitting the "Start"

button. JFLAP immediately gives a confirmation that the string is accepted by the

CNF grammar.

Editor	CYK Pa	rse							
Table Te	Table Text Size								
Star	t Step	Nonim	verted	I Tree					
Input <mark>pb</mark> String is	i/i/b/ppb/ s Accept	bi/ib/b/pp ed!	ib/b/i/p)					
LHS		RHS							
S	\rightarrow (CO							
s	$\rightarrow c$	CH	=	s					
s	$\rightarrow c$	CD	- Contraction						
S	\rightarrow	N							
A	\rightarrow F	M	000000						
м	\rightarrow E	BU	00000						
A	\rightarrow J	L	000000						
L	\rightarrow I	0	-						

After that, click on the Step button to see which production was first applied on the start symbol in the derivation. So, in the above picture, to derive the input string from our start symbol S, the first production that must be used is $S \rightarrow CN$.



After clicking Step as many times as necessary, the entire tree is obtained.

Editor	Editor CYK Parse							
Table Tex	t Size							
			\bigcirc					
Start Step Derivation Table								
String is /	/b/ppb	naidavavairia. Ipinipipibil	onib					
ou ing io i								
LHS		RHS		Production	Derivation			
S	\rightarrow	CO			S	-		
0		<u>с</u> ц		S→CN	CN			
3		СП	=	U→p N→AO	pin pAO			
S	\rightarrow	CD		A-B	p.RO			
0	\rightarrow	<u>CN</u>		J→b	pBRO			
3		CIN		R→IK	pbIKO			
A	\rightarrow	FM		I→FG	pbFGKO			
м	\rightarrow	BU		F→i	pbiGKO			
IVI	· ·	00	-	G→EF	pbiEFKO			
A	\rightarrow	JL		E→/	pbi/FKO			
1	\rightarrow			F→i	pbi/iKO			
	- <i>'</i>	162	-	K→EJ	рыледо			
В	\rightarrow	JW		E→/	phi/ijuu	_		
187	\rightarrow	KB	-	0→DP	nbiji/bDP			

The tree can be impossible to read if there are too many derivations, so an alternate method of viewing the productions used is to select 'Derivation Table' from the drop down menu next to the Start and Step buttons.

4.



First choose Pushdown Automaton from the start menu.



Next create the states for the automaton using the state creator.



Next create the transitions using the transition creator. When inputting the parameters for the transition arrows, the first parameter represents the input terminal being read in. The second parameter is the symbol being popped off the stack. The third parameter is the symbol being pushed onto the stack.



Here's what the PDA looks like after all the transition arrows have been added in.



However, JFLAP forces transition arrows to pop 1 symbol and push 0 or 2 symbols. This restriction forced me to change the PDA to the one pictured above.



The first step after entering the PDA into JFLAP is to click on the Convert menu and choose 'Convert to grammar'. After that you will arrive at the screen pictured above.



1000	LHS		RHS
0000000	(q0Zq0)	\rightarrow	(q1#q0)(q0Zq0)
00000000	(q0Zq0)	\rightarrow	(q1#q1)(q1Zq0)
00000000	(q0Zq0)	\rightarrow	(q1#q2)(q2Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q3)(q3Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q4)(q4Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q5)(q5Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q6)(q6Zq0)
	(q0Zq0)	\rightarrow	(q1#q7)(q7Zq0)
00000000	(q0Zq0)	\rightarrow	(q1#q8)(q8Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q9)(q9Zq0)
0000000	(q0Zq0)	\rightarrow	(q1#q10)(q10Zq0)
	(q0Zq0)	\rightarrow	(q1#q11)(q11Zq0)
00000000	(q0Zq0)	\rightarrow	(q1#q12)(q12Zq0)
0000000	(q0Zq1)	\rightarrow	(q1#q0)(q0Zq1)

Clicking on a transition adds all possible grammar rules associated with that transition. Because the grammar rules produced by this method were obtained using a brute force method, not all of them are useful.



To display all of the rules, click the show all button.

S	\rightarrow	LG
S	\rightarrow	ОН
S	\rightarrow	MA
С	\rightarrow	c
J	\rightarrow	c
U	\rightarrow	JK
U	\rightarrow	FC
F	\rightarrow	aFC
F	\rightarrow	aJK
F	\rightarrow	bIF
D	\rightarrow	BN
В	\rightarrow	aBN
В	\rightarrow	bIB
J	\rightarrow	Ud
G	\rightarrow	aUG
G	\rightarrow	aDA
L	\rightarrow	aFC
L	\rightarrow	aJK
М	\rightarrow	aBN
L	\rightarrow	bEL
0	\rightarrow	bEO
М	\rightarrow	bEM
0	\rightarrow	ε
G	\rightarrow	NA
E	\rightarrow	ε
Ι	\rightarrow	ε
Ν	\rightarrow	ε
В	\rightarrow	ε
A	\rightarrow	ε
Н	\rightarrow	ε
К	\rightarrow	ε

After that click export to get rid of useless grammar rules and display the final

grammar.

We know that $f(n) = \Omega(n^2)$. So $n^2 = O(f(n))$. This tells us that n^2 grows as f or slower. So, f grows as fast as n^2 or faster. Therefore we know that the string $w = 0^{n^2}$ is in the language. We can rewrite w as $w = (0^n)^n$. Now we make $w = (0^p)^p$ so that w is both in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| \le p$, |y| > 0, and xy^iz is recognized by the language for all $i \ge 0$. This means $x = 0^k$ and $y = 0^j$ where $k + j \le p$ and j > 0. Then take i = 0. This means that $xz = (0^{p \cdot j})^p$ is also in the language. However, since we know that j > 0, p - j must be less than p. So, $(p - j)p \le p^2$. This results in $(0^{p \cdot j})^p$ having fewer 0s than $(0^p)^p$. However, we know from before that f(n) must grow as fast as n^2 or faster. So, this string does not take the form $w = 0^{f(n)}$. This contradicts the pumping lemma and so the language is not regular.

b. Consider the string $w = a^p bca^p bc$. This string is in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| \le p$, |y| > 0, and $xy^i z$ is recognized by the language for all $i \ge 0$. This means $x = a^k$ and $y = a^j$ where $k + j \le p$ and j > 0. Then take i = 0. This means that $xz = a^{p-j}bca^p bc$ is also in the language. However, now there are fewer a's present on the left of the string. So, a couple of a's from the right half of the string will need to be moved to the left half to compensate for the smaller length. This causes the left half and right of the string to no longer to equivalent. So, it does not take the form ww. So, the string contradicts the pumping lemma and so the language is not regular.

5.

a.

c. Consider the string $w = 0^{p} \# 0^{4p} \# 0^{8p}$. This string is in the language and has a length greater than p. By the pumping lemma, let w = xyz. Also, $|xy| \le p$, $|xy| \le p$, $|xy| \le p$, |y| > 0, and $xy^{i}z$ is recognized by the language for all $i \ge 0$. This means $x = 0^{k}$ and $y = 0^{j}$ where $k + j \le p$ and j > 0. Then take i = 0. This means that $xz = 0^{p-j} \# 0^{4p} \# 0^{8p}$ is also in the language. However, since j > 0, we know that p - j is not equal to p. So, 4p does not equal 4(p - j) and 8p does not equal 8(p - j). So, the xz does have form $0^{p} \# 0^{4p} \# 0^{8p}$. Therefore, the string contradicts the pumping lemma and so the language is not regular.

6. The limerick that will be used is:
there once was a dinosaur
who decided to give up his roar,
but when he did
he flipped his lid
and started to roar even more!

rAux	Production Table	Hash
t	{}	8
th	{}	{th}
the	{}	{th, he}
ther	{}	{th, he, er}
there	{}	{th, he, er, re}
there_	{}	{th, he, er, re, e_}
there_o	{}	{th, he, er, re, e_, _o}
there_on	{}	{th, he, er, re, e_, _o, on}
there_once	{}	{th, he, er, re, e_, _o, on, nc}
there_once	{}	{th, he, er, re, e_, _o, on, nc, ce}
therAoncA	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA}
therAoncAw	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw}
therAoncAwa	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa}
therAoncAwas	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as}
therAoncAwas_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_}
therAoncAwas_a	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a}
therAoncAwas_a_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_}
therAoncAwas_a_d	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d}
		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d,
therAoncAwas_a_di	{A -> e_}	di}
		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d,
therAoncAwas_a_din	{A -> e_}	di, in}
		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d,
therAoncAwas_a_dino	{A -> e_}	di, in, no}
		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d,
therAoncAwas_a_dinos	{A -> e_}	di, in, no, os}
ther Approximation of dispace		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d,
therAoncAwas_a_dinosa	{A -> e_}	ui, iii, iiu, us, sa}
therAoncAwas a dinosau		$\{In, ne, er, rA, AO, On, nc, cA, AW, Wa, as, s_, a, a_, d, di in no os sa au\}$
inerAoncAwas_a_unosau	\X -> C_}	ui, iii, iio, os, sa, au
		(the baser rA. As an paseA. Awy was as a said
therAoncAwas a dinosaur	{A->e}	$\{i_1, i_2, e_1, i_2, \ldots, i_n, i_n, i_n, e_n, \ldots, w_n, w_n, a_n, s_n, s_n, a_n, a_n, \ldots, d_i i_n n_0 o_n s_n a_n, u_n\}$
	[/(> 0_]	$\int th he er r \Delta a an he c \Delta Aw we as s a a d$
therAoncAwas a dinosaur	{A -> e }	di, in. no. os. sa. au. ur. r }
	[, () 0_]	f the err A An on pc cA Aw wa as s a a d
therAoncAwas a dinosaur w	{A -> e }	di, in, no, os, sa, au, ur, r , w}
	·	{th, he, er, rA, Ao, on, nc, cA Aw wa as s a a d
therAoncAwas a dinosaur wh	{A -> e }	di, in, no, os, sa, au, ur, r , w, wh}
	·	{th, he, er, rA, Ao, on, nc, cA. Aw. wa. as. s . a. a . d.
therAoncAwas a dinosaur who	{A -> e }	di, in, no, os, sa, au, ur, r, w, wh, ho}
	r Aux t t th the ther there there there_on there_once there_once therAoncA therAoncAwa therAoncAwas_a therAoncAwas_a_d therAoncAwas_a_di therAoncAwas_a_dino therAoncAwas_a_dinos therAoncAwas_a_dinosau therAoncAwas_a_dinosau therAoncAwas_a_dinosau therAoncAwas_a_dinosau therAoncAwas_a_dinosau therAoncAwas_a_dinosau therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur therAoncAwas_a_dinosaur_w therAoncAwas_a_dinosaur_wh	r Aux Production Table t $\{\}$ the $\{\}$ the $\{\}$ there $\{\}$ t

_	therAoncAwas_a_dinosaur_who_	{A -> e_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, a_, _d, di, in, no, os, sa, au, ur, r_, _w, wh, ho, o_}
d	therAoncAwas aBinosaur whoB	{A->e,B-> d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r, w, wh, ho, oB}
		, _,	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
е	$ther Aonc Awas_a Binos aur_who Be$	{A -> e_, B -> _d}	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be}
	therAoncAwas_aBinosaur_whoBe		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
С		{A -> e_, B -> _d}	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec}
i	ci	{A -> e_, B -> _d}	{tn, ne, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci}
	therAoncAwas_aBinosaur_whoBe		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
d	cid	{A -> e_, B -> _d}	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id}
	therAoncAwas aBinosaur whoBe		{tn, ne, er, rA, Ao, on, nc, cA, AW, Wa, as, s_, _a, aB, Bi, in no os sa au ur r w wh ho oB Be ec ci id
е	cide	{A -> e_, B -> _d}	de}
			$\{th,he,er,rA,Ao,on,nc,cA,Aw,wa,as,s_,_a,aB,Bi,$
d	therAoncAwas_aBinosaur_whoBe		in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
a	cided	{A -> e_, B -> _0}	$\{de, eu\}$
	therAoncAwas aBinosaur whoBe		in, no, os, sa, au, ur, r , w, wh, ho, oB, Be, ec, ci, id,
_	cided_	{A -> e_, B -> _d}	de, ed, d_}
			{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
	therAoncAwas_aBinosaur_whoBe		in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
ι	cided_t	{A -> e_, D -> _u}	$\{t, eu, u_{, -}, t\}$
	therAoncAwas aBinosaur whoBe		in. no. os. sa. au. ur. r . w. wh. ho. oB. Be. ec. ci. id.
0	cided_to	{A -> e_, B -> _d}	de, ed, d_, _t, to}
			$\{ th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, $
	therAoncAwas_aBinosaur_whoBe		in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
_	cided_to_	{A -> e_, B -> _0}	$de, ed, d_{,-}[, to, _0]$
	therAoncAwas aBinosaur whoBe		in, no, os, sa, au, ur, r , w, wh, ho, oB, Be, ec, ci, id,
g	cided_to_g	{A -> e_, B -> _d}	de, ed, d_, _t, to, _o, _g}
			{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
:	therAoncAwas_aBinosaur_whoBe		in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
I	cided_to_gi	{A -> e_, D -> _u}	$de, ed, d_{,} _{l}, l, lo, _0, _0, g, g$
	therAoncAwas aBinosaur whoBe		in. no. os. sa. au. ur. r . w. wh. ho. oB. Be. ec. ci. id.
v	cided_to_giv	{A -> e_, B -> _d}	de, ed, d_, _t, to, _o, _g, gi, iv}
			{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
	therAoncAwas_aBinosaur_whoBe		in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
е	cided_to_give	{A -> e_, B -> _0}	$de, ed, d_{,} _{l}, to, _o, _g, gl, iv, ve$
	therAoncAwas aBinosaur whoBe	{A -> e marked,	in, no, os, sa, au, ur, r , w, wh, ho, oB, Be, ec, ci, id,
_	cided_to_givA	B -> _d}	de, ed, d_, _t, to, _o, _g, gi, iv, vA}
			{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
	therAoncAwas_aBinosaur_whoBe	$\{A \rightarrow e_marked, B \rightarrow d\}$	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,
u		□ -> _u}	(the errA Acon nc cA Aw wa as so a a R Ri
	therAoncAwas_aBinosaur whoBe	{A -> e_ marked.	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id.
р	cided_to_givAup	B -> _d}	de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up}
	therAoncAwas_aBinosaur_whoBe	{A -> e_ marked,	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi,
_	cided_to_givAup_	B -> _d}	in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id,

			de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_}
h	therAoncAwas_aBinosaur_whoBe cided_to_givAup_h	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h}
i	therAoncAwas_aBinosaur_whoBe cided_to_givAup_hi	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi}
S	therAoncAwas_aBinosaur_whoBe cided_to_givAup_his	{A -> e_ marked, B -> _d}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, as, s_, _a, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, is}
_	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiC	{A -> e_ marked, B -> _d, C -> s_}	$\{$ th, he, er, rA, Ao, on, hc, cA, Aw, wa, aC, Ca, aB, Bi, In, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC $\}$
r	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCr	{A -> e_ marked, B -> _d, C -> s_}	(iii, he, ei, iA, Ao, oii, hc, cA, Aw, wa, aC, Ca, aB, Bi, iii, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr} {th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,
0	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCro	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro}
а	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroa	{A -> e_ marked, B -> _d, C -> s_}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa}
r	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar	{A -> e_ marked, B -> _d, C -> s_}	in, he, ei, iA, Ao, on, he, cA, Aw, wa, ac, Ca, ab, bi, in, no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar} $\{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, ic, ch, ab, ab, ab, ab, ab, ab, ab, ab, ab, ab$
,	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , } {th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,
_	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,_	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _} /th be er rA Ao on pc cA Aw wa aC Ca aB Bi in
b	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,_b	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , ,_, _b} /th be er rA Ao on nc cA Aw wa aC Ca aB Bi in
u	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,_bu	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r ,_, _b, bu}
t	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,_but	{A -> e_ marked, B -> _d, C -> s_}	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut} $\{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in$
_ W	therAoncAwaCaBinosaur_whoBe cided_to_givAup_hiCroar,_but_ therAoncAwaCaBinosaurDhoBeci	{A -> e_ marked, B -> _d, C -> s_} {A -> e_ marked,	no, os, sa, au, ur, r_, _w, wh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, t_} {th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,

	ded_to_givAup_hiCroar,_butD	B -> _d, C -> s_, D -> _w}	no, os, sa, au, ur, rD, Dh, ho, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tD}
h	therAoncAwaCaBinosaurEoBecid ed_to_givAup_hiCroar,_butE	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE}
е	therAoncAwaCaBinosaurEoBecid ed_to_givAup_hiCroar,_butEe	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee} {th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,
n	therAoncAwaCaBinosaurEoBecid ed_to_givAup_hiCroar,_butEen	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en} {th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in,
_	therAoncAwaCaBinosaurEoBecid ed_to_givAup_hiCroar,_butEen_	{A -> e_ marked, B -> _d, C -> s_, D -> _w, E -> Dh}	no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, p_, _h, hi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, n_}
h	therAoncAwaCaBinosaurEoBecid ed_to_givAupFiCroar,_butEenF	$\begin{array}{l} \{A \Rightarrow e_marked, \\ B \Rightarrow _d, C \Rightarrow s_, \\ D \Rightarrow _w, E \Rightarrow Dh, \\ F \Rightarrow _h \end{array}$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nF}
е	therAoncAwaCaBinosaurEoBecid ed_to_givAupFiCroar,_butEenFe	$\begin{array}{l} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h \} \end{array}$	$ \{ th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_,t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , b, bu, ut, tE, Ee, en, nF, Fe \} $
_	therAoncAwaCaBinosaurEoBecid ed_to_givAupFiCroar,_butEenFA	$\begin{array}{l} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h \} \end{array}$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nF, FA}
d	therAoncAwaCaBinosaurEoBecid ed_to_givAupFiCroar,_butEenFA d	$\begin{array}{l} \{A \mathrel{\to} e_marked, \\ B \mathrel{\to} _d, C \mathrel{\to} s_, \\ D \mathrel{\to} _w, E \mathrel{\to} Dh, \\ F \mathrel{\to} _h \end{array}$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, ci, id, de, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nF, FA, Ad}
i	therAoncAwaCaBinosaurEoBecid ed_to_givAupFiCroar,_butEenFA di	$\begin{array}{l} \{A \mathrel{\rightarrow} e_marked, \\ B \mathrel{\rightarrow} _d, C \mathrel{\rightarrow} s_, \\ D \mathrel{\rightarrow} _w, E \mathrel{\rightarrow} Dh, \\ F \mathrel{\rightarrow} _h \end{array}$	
d	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenFA dG	$ \begin{array}{l} \{A \mathrel{\rightarrow} e_marked, \\ B \mathrel{\rightarrow} _d, C \mathrel{\rightarrow} s_, \\ D \mathrel{\rightarrow} _w, E \mathrel{\rightarrow} Dh, \\ F \mathrel{\rightarrow} _h, G \mathrel{\rightarrow} id \\ \end{array} $	
_	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenFA dG_	$ \begin{array}{l} \{A \mathrel{\rightarrow} e_marked, \\ B \mathrel{\rightarrow} _d, C \mathrel{\rightarrow} s_, \\ D \mathrel{\rightarrow} _w, E \mathrel{\rightarrow} Dh, \\ F \mathrel{\rightarrow} _h, G \mathrel{\rightarrow} id \\ \end{array} $	
h	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenFA dGF	$\begin{array}{l} \{A \mathrel{\rightarrow} e_marked, \\ B \mathrel{\rightarrow} _d, C \mathrel{\rightarrow} s_, \\ D \mathrel{\rightarrow} _w, E \mathrel{\rightarrow} Dh, \\ F \mathrel{\rightarrow} _h, G \mathrel{\rightarrow} id \} \end{array}$	

е	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenFA dGFe	$\begin{array}{l} \mbox{$\{A = > e_marked, $B = > _d, $C = > s_, $, $D = > _w, $E = > Dh, $F = > _h, $G = > id$} \end{array}$	
_	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GH	$\begin{array}{l} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h, G \rightarrow id, \\ H \rightarrow FA \} \end{array}$	$ \{ th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH \} $
f	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHf	$\begin{array}{l} $\{A \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf}
I	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHfl	$\begin{array}{l} \label{eq:alpha} \{A \ -> \ e_{} \ marked, \\ B \ -> \ d, \ C \ -> \ s_{}, \\ D \ -> \ w, \ E \ -> \ Dh, \\ F \ -> \ h, \ G \ -> \ id, \\ H \ -> \ FA \end{array}$	$ \{ th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl \} $
i	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHfli	$\begin{array}{l} \label{eq:alpha} \{A \ -> \ e_{} \ marked, \\ B \ -> \ d, \ C \ -> \ s_{}, \\ D \ -> \ w, \ E \ -> \ Dh, \\ F \ -> \ h, \ G \ -> \ id, \\ H \ -> \ FA \end{array}$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li}
р	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHflip	$\begin{array}{l} \label{eq:alpha} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h, G \rightarrow id, \\ H \rightarrow FA \} \end{array}$	{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip}
р	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHflipp	$\begin{array}{l} \label{eq:alpha} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h, G \rightarrow id, \\ H \rightarrow FA \} \end{array}$	
е	therAoncAwaCaBinosaurEoBecG ed_to_givAupFiCroar,_butEenHd GHflippe	$\begin{array}{l} \label{eq:alpha} \{A \rightarrow e_marked, \\ B \rightarrow _d, C \rightarrow s_, \\ D \rightarrow _w, E \rightarrow Dh, \\ F \rightarrow _h, G \rightarrow id, \\ H \rightarrow FA \} \end{array}$	$ \{ th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, Ge, ed, d_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip, pp, pe \} $
d	therAoncAwaCaBinosaurEoBecG J_to_givAupFiCroar,_butEenHdG HflippJ		{th, he, er, rA, Ao, on, nc, cA, Aw, wa, aC, Ca, aB, Bi, in, no, os, sa, au, ur, rE, Eo, oB, Be, ec, cG, GJ, J_, _t, to, _o, _g, gi, iv, vA, Au, up, pF, Fi, iC, Cr, ro, oa, ar, r , _, _b, bu, ut, tE, Ee, en, nH, Hd, dG, GH, Hf, fl, li, ip, pp, pJ}

In the table above, spaces in the sentences were replaced by $\underline{\ }$ for clarity.

The grammar outputted for this would be:

{A -> e_,
B -> _d,
C -> s_,
D -> _w,
E -> Dh,
F -> _h,
G -> id,
H -> FA,
J -> ed,
S -> therAoncAwaCaBinosaurEoBecGJ_to_givAupFiCroar,_butEenHdGHflippJ}

The final compressed string would be:

Re_R_dRs_R_wR#3hR_hRidR#5#0Red (continued on next line)

Rther#0onc#0wa#2a#1inosaur#4o#1ec#6#8_to_giv#0up#5i#2roar,_but#4en#7d#6#7flipp#8