(H) Sequitur (1/2) [Solution]

The Sequitur algorithm was introduced in 1997 by Craig Nevill-Manning and Ian Witten to "identify hierarchical structure in sequences." Note that the version of Sequitur shown in this problem has been modified slightly from the original version (specifically, the "rule utility" property is not enforced in this problem). Byte pair encoding (BPE) was described in a 1994 paper by Philip Gage. It has been used in many of the most sophisticated natural language processing systems developed in recent years.

H1. The complete table for Sequitur is:

Step Number	Column 1	Column 2
1)	a	S = a
2)	ab	S = ab
3)	abc	S = abc
4)	**(a)** abcd	S = abcd
5)	abcdb	**(b)** S = abcdb
6)	abcdbc	S = abcdbc
	aXdX	S = aXdX
		X = bc
7)	**(c)** aXdXa	**(d)** S = aXdXa
		(e) X = bc
8)	aXdXab	**(f)** S = aXdXab
		X = bc
9)	aXdXabc	S = aXdXabc
		X = bc
	aXdXaX	S = aXdXaX
		(g) X = bc
	(h) YdXY	**(i)** S = YdXY
		X = bc
10)	**(j)** YdXYd	Y = aX
		(k) S = YdXYd
	ZXZ	X = bc
		Y = aX
		S = ZXZ
		X = bc
		Y = aX
S = ZXZ; X = bc; Y = aX; Z = Yd		

Brief explanation: Sequitur considers its input letter by letter, moving left-to-right. When it encounters a two-letter-sequence that it has seen before, it replaces all instances with a new letter ("nonterminal", e.g., X), and makes any other possible replacements of repeating sequences with nonterminals. Column 1 records the portion of the input under consideration at each step (including the results of nonterminal replacements) and Column 2 keeps track of the nonterminal-to-letter correspondences in replacements.

(H) Sequitur (2/2) [Solution]

H2. The complete table for BPE is:

Step Number	Column 1	Column 2
1)	abcdbcabcd	S = abcdbcabcd
2)	**(a)** aXdXaXd	**(b)** S = aXdXaXd X = bc
3)	YdXYd	S = YdXYd X = bc **(c)** Y = aX
4)	ZXZ	S = ZXZ X = bc Y = aX Z = Yd
S = ZXZ; X = bc; Y = aX; Z = Yd		

Brief explanation: BPE considers the entire input at once, and determines the most-commonly-occurring two-letter sequence across the whole input (breaking ties in favour of the leftmost sequence). This sequence is replaced with a nonterminal, and the process repeats, continuing until no replacements are possible. Columns 1 and 2 serve purposes analogous to those in Sequitur.

H3.

(a) Different — *Sequitur* output: S = XcXdYY; X = ab; Y = bc
BPE output: S = aXabdXX; X = bc

(b) Same — *Sequitur* and *BPE* output: S = abbXddX; X = ca

(c) Different — *Sequitur* output: S = VV; X = cb; Y = ba; Z = YX; W = ZX; V = Wa
BPE output: S = VV; X = ba; Y = Xc; Z = Yb; W = Zc; V = WX

Note: For symbols after Z, we use W, then V, and so on (going backwards), but any choices of new, distinct symbols are consistent with the data shown.

(d) Same — *Sequitur* and *BPE* output: S = ZZXaX; X = cc; Y = Xd; Z = Yb

(e) Different — *Sequitur* output: S = ZZWWW; X = cc; Y = Xd; Z = Yb; W = Xa
BPE output: S = WWYYY; X = cc; Y = Xa; Z = Xd; W = Zb

Note: since these two outputs have the same "structure", that is, they are differentiated only by the specific nonterminal symbols used, the answer "Same" was given partial credit. Also, it is consistent with the data to suppose that both algorithms will make one more compression, on the final three characters (e.g., Sequitur might end with S = WWVY, with V = YY). This will not affect the answer here.