1. This can yield a string with no b’s.
2. This won’t generate the string abaaaba.
3. The union symbol means choose any of the options, not combine them.
4. Don’t put lambdas in the answer; they’re not considered “characters” in a string.
5. This can generate the string aabba.
6. This should be split into multiple steps.
7. You need to put S -> at the beginning of every derivation.
8. These can yield strings with an even number of characters.
9. This won’t generate the string abbbaa.
10. This will not guarantee the number of b’s equal the number of a’s.
11. This won’t generate the string bba.
12. What is this symbol? Minus? It doesn't belong here.
13. A derivation looks like this: S > ASB > aAbSb etc. You were drawing derivation trees.
14. You’ve got the right idea, but two odds make an even. You need two evens and one odd.
15. This doesn’t guarantee the string will start and end with a’s.
16. This won’t generate the string abbaa.
17. This is not a rightmost derivation.
18. This is not a leftmost derivation.
19. This can generate the string aabbbaba which is not in the language.
20. The intersection symbol should not be used here.
21. This is the same as {a,b}*.
22. This will allow more than 2 b’s.
23. A cannot yield aBb.
24. This creates an infinite loop.
25. Also auaua.
26. This isn’t necessary.
27. It’s true that you must have one or more a’s, but your +’s indicate there must be three or more a’s.
28. This doesn’t allow for the string aba.
29. This doesn’t allow for a string of all a’s.
30. This allows abb.
31. This won’t generate the string aba.
32. There is a much simpler way to do this.
33. There are two interpretations of this question: 1. All a’s precede the b’s which precede the c’s and the empty string is included, but there can be b’s without a’s and c’s without b’s. The answer to this interpretation is a*b*c*. 2. All a’s precede all b’s which precede all c’s and the empty string is included, but there cannot be any b’s with at least one a, and there cannot be any c’s without at least one b (and thus also at least one a). The answer to this interpretation is a* U a+b* U a+b+c*.
34. This won’t generate the string abba.
35. Your regular expression allows b+, which is not in the language.
36. This requires every a be followed by a b.
37. Including the substrings aba and bab satisfies the condition that ab and ba are substrings, but your grammar doesn't allow for that.
38. This doesn't allow for the strings aba and bab.
39. There do not need to be the same number of a's as b's.
40. All a's must precede all b's.
41. There have to be more pairs of a's than there are single b's.
42. The language you describe here is made up *only* of sets of the substrings ab and ba. You need to include strings such as aaaababaaaaabbbbbbb.
43. Based on your definition, this language must include *both* the substrings ba and ab.
44. This only allows for strings of length 2.
45. This doesn’t allow for any a’s by themselves, only with b’s.
46. This language does not need to start with aa.

47. This is not what a derivation tree looks like.
48. This is not set notation.
49. This is not a regular grammar.
50. The grammar you describe only allows for 2 b's and having a* allows any number of a's, meaning there can be some that are not preceded or followed by a b.
51. You don't need the same number of a's in the front as in the back, but you do need a group of (ab)'s followed by a group of (ba)'s.
52. Your grammar won't allow a string that starts with a b or contains bab.
53. The grammar described by the problem allows a's without b's, b's without c's, etc. Yours does not.