Problem 6.1 [Reduction Properties]. Suppose you have a polynomial-time one-call reduction \( r \) from NP search problem \( A \) to NP search problem \( B \), which converts each problem-\( A \) instance \( x \) having \( a \) solutions into a problem-\( B \) instance \( r(x) \) having \( f(x, a) \) solutions.

(a) What must be true about \( f \) for \( r \) to preserve NP-hardness (i.e., \( A \) being NP-hard implies \( B \) is NP-hard)?

(b) What must be true about \( f \) for \( r \) to preserve \#P-hardness (i.e., \( \#A \) being \#P-hard implies \( \#B \) is \#P-hard)?

(c) What must be true about \( f \) for ASP \( A \) being NP-hard to imply ASP \( B \) is NP-hard?

(d) Given an NP search problem \( A \), define \( \text{AASP} \ A \) to be the following decision problem: given an instance of \( A \) and two distinct solutions of that instance, is there a third distinct solution? What must be true about \( f \) for \( \text{AASP} \ A \) being NP-hard to imply \( \text{AASP} \ B \) is NP-hard?

(e) Besides the function \( r \) converting instances of \( A \) into instances of \( B \), we require another function to conclude that ASP \( B \) is NP-hard. What must the other function do? What about for AASP?