Lambda calculus for programming constructs

1. In the basic untyped lambda calculus, the boolean “true” is encoded as λx. λy. x, and “false” is encoded as λx. λy. y. That is, “true” takes in two arguments and returns the first, while “false” takes in two arguments and returns the second. These definitions of the boolean constants may seem strange, but they are designed to work with the “if-then-else” expression. The if-then-else expression is defined as λx. λy. λz. ((x y) z). Verify that these definitions do, indeed, make sense, by evaluating the following:
    a. (((λx. λy. λz. ((x y) z)λu. λv. u) A) B)
    b. (((λx. λy. λz. ((x y) z)λu. λv. v) A) B)

Ocaml

2. Suppose a weighted undirected graph (where each vertex has a string name) is represented by a list of edges, with each edge being a triple of the type String * String * int. Write an OCaml function to identify the minimum-weight edge in this graph. Use pattern matching to solve this problem.

3. Solve the above problem by using the List.fold_left higher-order function.