1995 Paper 1 Question 3

Foundations of Computer Science

This question has been translated from Standard ML to OCaml

An OCaml list can be considered to be a set if it has no repeated elements, e.g., `[4; 2; 3]` is a set but `[4; 2; 4; 3]` is not.

Write an OCaml function `intersect` to compute the set-theoretic intersection of two lists that satisfy this property of being a set. (The intersection of two sets is the set of elements that appear in both sets.) Your function must also satisfy conditions (a)–(c) below:

(a) The result list has no repeated elements;

(b) The number of cons (`::`) operations performed does not exceed the number of elements in the result list;

(c) The elements of the result list appear in the same order as they do in the first argument list.

You may assume the existence of `[]` (empty list) and `::` (cons operation). All other functions over lists must be defined by you. Little credit will be given for answers that do not satisfy conditions (a)–(c).

Write an OCaml function `subtract` that given two lists satisfying the property of being a set, returns a list consisting of those elements of the first list that do not appear in the second list. Your subtract function must satisfy conditions (a)–(c) above.

Write an OCaml function `union` that given two lists satisfying the property of being a set, returns a list consisting of those elements that appear in one or other or both of the lists. Your union function must satisfy conditions (a) and (b) above and *(d)* below:

(d) Elements from the first argument list appear before any others in the result and in the same order as they appear in the first argument.

State the most general type of your union function.

[10 marks]