INTRO. TO LOGIC & FUNCT. PROG.

This functional programming assignment requires you to represent finite sets (of any arbitrary type 'a)

(a) using OCaml lists.

Representation invariant: a set is represented as a list without duplicates.

You need to implement the following operations:

1. **emptyset**, which represents the empty set.
2. **member** *x s*, which returns true if and only if *x* is in *s*.
3. **cardinality** *s,* which returns the number of elements in the set *s.*
4. **union** *s1 s2*, which returns the union of sets *s1* and *s2*
5. **intersection** *s1 s2*, which returns the intersection of *s1* and *s2*
6. **difference** *s1 s2*, which returns the set consisting of elements of *s1* which are not in *s2*
7. **product** *s1 s2*, which returns the cartesian product of *s1* and *s2*.
8. **power** *s*, which returns the set of subsets of *s.*
9. **subset** *s1 s2*, which returns true if and only if *s1* is a subset of *s2.*
10. **equalset** *s1 s2*, which returns true if and only if set *s1* is equal to set *s2*.

Wherever possible, use the list functions **map, filter, fold**, etc.

In your documentation, you need to show that if the input set(s) satisfy the Representational Invariant, then so do the sets returned by the operations **emptyset, union, intersection, difference, product, power.**

You will also need to show that for example the following laws about union:

for all *x, s1, s2*:

1. **member** *x* **emptyset** = false
2. **cardinality emptyset** = 0
3. **member *x s1* implies member *x (union s1 s2)***
4. ***member x (intersection s1 s2) implies member x s1***
5. ***equalset*** *(intersection s1 s2)* (***intersection*** *s2 s1)*
6. ***cardinality*** (***product*** *s1 s2)* = ***cardinality s1 * cardinality s2***

7. **...and other such laws**

(b) Consider now representing a set i by its characteristic function [Recall that *f_s* is the characteristic function of set *s* when *x* \in *s* iff *f_s (x)* = true]

You need to implement the following operations:

1. **emptyset**, which represents the empty set.
2. **member** *x s*, which returns true if and only if *x* is in *s*.
3. **union** *s1 s2*, which returns the union of sets *s1* and *s2*
4. **intersection** *s1 s2*, which returns the intersection of *s1* and *s2*
5. **difference** *s1 s2*, which returns the set consisting of elements of *s1* which are not in *s2*
6. **product** *s1 s2*, which returns the cartesian product of *s1* and *s2*.

Again, in your documentation show that the result of any operation is the characteristic function of the set being represented.