CSE 341 - Section 2 Exercises (from website code)

1. Type synonyms

Below is an example use of type synonyms, with three example date representations:

```
type date = int * int * int;
val date1 : int * int * int = (22, 2, 1900);
val date2 : date = (11, 1, 1900);
val date3 = (17, 1, 2019);
```

A. Without using this type, write a function `dmy_to_mdy` to change the date format from (day, month, year) to (month, day, year). Your function should return a triple.

B. Now write `dmy_to_mdy2` using the `date` type synonym instead.

C. For each of your two functions, write test calls using the example dates which are valid arguments.

2. Type Generality and Equality Types

A. Review: Write a function `append` that takes two string list arguments xs and ys and returns the result of appending them together.

B. Now change your function to accept any 'a list and returns an appended result 'a list. This function should return the same list for the example calls in 2.A.

C. Now consider the following lists:
Consider the following lists:

```
val list1 = ["hi", "bye"];
val list2 = ["programming", "languages"];
val list3 = [1, 2];
val list4 = [3, 4, 1];
```

Write three test calls to your function in B - two should be correct (and provide the result) and the third should result in a type error.

D. (Equality types): Write a function `contains` which takes as argument an element x and a list of elements xs and returns true if and only if x is contained in xs. Then write out the type of the function as a comment header.

E. (Equality types). Consider the following functions. What are each of their types, as printed in the REPL?

```
(* ________________________________________ *)
fun is_three(x, y) = if x = 3 then "yes" else "no"

(* ________________________________________ *)
fun same_thing(x, y) = if x = y then "yes" else "no"
```

3. More datatype and Pattern-matching examples

Consider the following type and datatype:

```
type cart = real * real
datatype shape = Circle of cart * real (* coordinates and radius *)
               | Square of cart * real (* coordinates and side length *)
               | Rectangle of cart * real * real (* coordinates and side lengths *)
```

A. Write a function `area` which takes a shape as an argument and returns its area (as a real value).

Now recall this datatype to represent expression trees from lecture:

```
datatype exp = Constant of int
             | Negate   of exp
             | Add      of exp * exp
             | Multiply of exp * exp
```

B. Write an ML function `const_not_under_add` of type `exp -> bool` that returns true if and only if there exists a Constant in the expression that is not a child of an Add expression. For example, `const_not_under_add(Constant 341)` should return true, as should `const_not_under_add(Multiply(Constant 341, Add(Constant 0, Constant 1)))`.

C. On your own: What is another function you can think of using the shape or exp datatypes?