Options

Options are alot like lists, but instead of containing an arbitrary amount of elements, they contain either one element or NONE, and just like lists, options are built and accessed.

`t option` is a type for any type `t`

Building an Option

`NONE` builds an option that contain 0 elements that has a type of `a option`

`SOME e` builds an option that contains 1 element you evaluate it by evaluating `e` It has the type `t option` if `e` has `type t`

Accessing an Option

We need to access our options

`isSome`: is a function that has type of `'a option -> bool` it returns `true` if its a `SOME e` and false if its a `NONE`. A lot like null for lists

valOf has a type `'a option -> 'a` takes an option and gets the `e` out from underneath the `SOME` it raises an `exception` if the argument is `NONE`

Example of using an Option

```
fun better_max2 (xs : int list) =
    if null xs
    then NONE
    else
        let (* ok to assume xs nonempty b/c local *)
            fun max_nonempty (xs : int list) =
                if null (tl xs)
                then hd xs
                else
                    let val tl_ans = max_nonempty(tl xs)
                    in
                        if hd xs > tl_ans
                        then hd xs
                        else tl_ans
                    end
    in
        SOME (max_nonempty xs)
end
```