Exercises with lists

- **Exercise 1: Working with lists.** Implement a function `rev` of type `rev : list<'a> -> list<'a>` which reverses a list.

```
> rev [1;2;3];;
val it : int list = [3; 2; 1]
```

What is the runtime complexity of `rev` (in Big O Notation)?

- **Exercise 2: Parallel list traversal.** Implement a function `equalBy` of type `equalBy : ('a -> 'a -> bool) -> list<'a> -> list<'a> -> bool` which compares the element-wise equality of two lists using a predicate function. The first argument is the function used for element-wise comparison. If the function returns true for *all* elements, the overall result is true, otherwise false.

```
> equalBy (=) [1;2;3] [1;2;3];;
val it : bool = true

> equalBy (=) [1;2;3] [1;2;10];;
val it : bool = false

> equalBy (=) [1;2;3] [1;2];;
val it : bool = false
```

(*The binary function `(=)` checks two things for equality in F#. The equivalent lambda is `fun a b -> a = b`*.)

- **Exercise 3: Thinking declaratively.** Implement a function `isPalindrome` of type `isPalindrome : ('a -> 'a -> bool) -> list<'a> -> bool`, which checks whether the list forms a palindrome (using the provided element-wise comparison function in the first argument). A palindrome is a list that is equal to itself reversed.

  Think in a declarative manner. The function from (1) reverses a list. The function from (2) checks if two lists are equal.

```
> isPalindrom (=) [’a’; ’b’; ’c’];;
val it : bool = false
> isPalindrom (=) [’a’; ’b’; ’b’; ’a’ ];;
val it : bool = true
```

The use of a *higher order function* lets us do less strict comparison:

```
let cmpInsensitive a b =
    System.Char.ToLower a = System.Char.ToLower b

> isPalindrom cmpInsensitive ['a';'B';'A'] ;;
val it : bool = true
```

- **Exercise 4: Recursion.** Define a function `mapi` of type `mapi : (int -> 'a -> 'b) -> list<'a> -> list<'b>` which maps each element's zero based index `int` and its value `'a` to a new value of type `'b`.

```
> mapi (fun i e -> (i,e)) ['a';'b';'c'];;
val it : (int * char) list = [(0, 'a'); (1, 'b'); (2, 'c')]
```

(*The function* `List.mapi` *can also be found in the F# base libraries. Your solution should behave identically. The documentation can be found here: https://tinyurl.com/y3ck7az2 .*)