How to obtain iteration from recursion?

- Given a list L and an integer i write a function $CycleBis$ that cycles L i times. Given a list $[a_1, ..., a_n]$ this function produces $[a_{i+1}, a_{i+2}, ..., a_n, a_1, a_2, ..., a_i]$.

```
Precondition for CycleBis: i is >= 0.

fun CycleBis (L,0) = L
| CycleBis(L,i) = CycleBis(Cycle(L),i-1);

val CycleBis = fn : 'a list * int -> 'a list
```

Example1

- Write a function *IsEmptyList* that tests if a list is empty using pattern-matching.

```
fun IsEmptyList([]) = true
| IsEmptyList(L) = false;

val IsEmptyList = fn : 'a list -> bool
```

- Write a function *Cycle* that cycles a list once. Given a list $[a_1, ..., a_n]$ this function produces $[a_2, a_3, ..., a_n, a_1]$.

```
fun Cycle [] = []
| Cycle(x::L) = L @ [x];

val Cycle = fn : 'a list -> 'a list
```

Example 2

- Write a function $CopyDel$ that given a list L and an integer i returns a copy of L without the i^{th} element.

```
We assume the first element of a list is at position 1.

Precondition for CopyDel1: j <= i.

fun CopyDel1(L, j, i) =
if L = []
then []
else if i = j
     then tl(L)
     else hd(L)::CopyDel1(tl(L),j+1,i);

fun CopyDel(L,i)=
if (i > length(L)) orelse (i <= 0)
then L
else
    if L=[]
    then []
    else CopyDel1(L,1,i);

val CopyDel = fn : 'a list * int * int -> 'a list
```

```
CopyDel([],5);            CopyDel([2],1);
CopyDel([2],5);           CopyDel([2,3],0);
CopyDel([2,3],1);         CopyDel([2,3],2);
CopyDel([1,4,6,7,8],1);   CopyDel([1,4,6,7,8],2);
CopyDel([1,4,6,7,8],5);   CopyDel([1,4,6,7,8],0);
```