

Higher-order functions

Question 1: Define a function `swapargs` which takes a function `f` of type `'a * 'b -> 'c` and returns a new function of type `'b * 'a -> 'c` such that $f(x, y) = \text{swapargs}(f)(y, x)$ for all `x` and `y`.

Question 2: Define the functions

1. `curry : ('a * 'b -> 'c) -> 'a -> 'b -> 'c`
2. `uncurry : ('a -> 'b -> 'c) -> 'a * 'b -> 'c`

`curry` takes a function `f : 'a * 'b -> 'c` and gives back the curried function `g : 'a -> 'b -> 'c` such that $g\ x\ y = f(x, y)$. `uncurry` does the reverse, i.e. takes a function like `g` and returns a function like `f`.

Question 3: Define a function `nfold` which takes two arguments, `f` and a non-negative integer `n`, and returns the `n`-fold composition of `f`. Namely, $\text{nfold}(f, n)(x) = f^n(x)$. Note that $f^0(x) = x$.

Question 4: Define a function `fixedpoint` which takes two arguments, a function `f` and an element `x`, and returns the smallest positive integer `n` such that $f^n(x) = x$. Note that if the function has no fixed point, then this operation won't terminate. (I don't recommend using `nfold` for this, but you may.)

Question 5: Consider the function `reduce: ('a * 'b -> 'b) -> 'b -> 'a list -> 'b` defined as

```
fun reduce f v [] = v
  | reduce f v (x::l) = f(x, reduce f v l);
```

Using `reduce`, define the functions

1. `exists 'a list -> ('a -> bool) -> bool` such that `exists l p` holds if and only if in the list `l` there is at least an element which satisfy the property `p`.
2. `forall 'a list -> ('a -> bool) -> bool` such that `forall l p` holds if and only if every element of the list `l` satisfies the property `p`.

The definitions should have the form

```
fun exists l p = reduce ...;
fun forall l p = reduce ...;
```