Homework 4: ML Pattern Matching

Stephen Wagstaff
CS 431
October 4, 2018

---

1. Write a function `zip` that takes two lists and return a list of 2-tuples.
   - For example, `zip ([1, 2, 3], [4, 5])` should return `[(1,4), (2,5)]`. Note that if one list is longer than the other, the unmatched portion of the longer list is ignored.

```
fun zip (nil, _) = nil
  | zip (_, nil) = nil
  | zip (h1::list1, h2::list2) = (h1, h2) :: zip (list1, list2);
```

2. Write a function `unzip` that takes a list of 2-tuples and return a tuple of two lists. -For example, `unzip [(1,2), (3,4), (5,6)]` should return `([1,3,5], [2,4,6])`

```
fun unzip(nil) = (nil, nil)
  | unzip((leftElement, rightElement)::list) =
  let
    val (leftList, rightList) = unzip(list)
  in
    (leftElement::leftList, rightElement::rightList)
  end;
```

3. Write a function `zip3` that takes three lists and return a list of 3-tuples.
   - For example, `zip3 ([1, 2, 3], [4, 5], [6,7,8])` should return `[(1,4,6), (2,5,7)]`. Note that if one list is longer than the others, the unmatched portion ofthe longer list is ignored.

```
fun zip3 (nil, _, _) = nil
  | zip3 (_, nil, _) = nil
  | zip3 (_, _, nil) = nil
  | zip3 (h1::list1, h2::list2, h3::list3) = (h1, h2, h3) :: zip3 (list1, list2, list3);
```

4. Write a function `unzip3` that takes a list of 3-tuples and return a tuple of three lists.
   - For example, `unzip3 [(1,2,3), (4,5,6), (7,8,9)]` should return `([1,4,7], [2,5,8], [3,6,9])`.

```
fun unzip3(nil) = (nil, nil, nil)
  | unzip3((leftElement, centerElement, rightElement)::list) =
  let
    val (leftList, centerList, rightList) = unzip3(list)
  in
      (leftElement::leftList, centerElement::centerList, rightElement::rightList)
  end;
```

5. Write a function `zipWithIndex` that takes a list and return a list of 2-tuples, where each tuple contains an index and a list element.
   - For example, `zipWithIndex ["a", "b", "c"]` should return `[(0, "a"), (1, "b"), (2, "c")]`.

```
fun zipWithIndex (list) =
  let
    fun zipWithIndexHelper(nil, _) = nil
      | zipWithIndexHelper(x::list, counter) = (counter, x)::zipWithIndexHelper(list, counter+1)
  in
    zipWithIndexHelper(list, 0)
  end;
```

6. Write a function `flatten` that takes a list of lists and return a flat-tened list.
   - For example, `flatten [[1,2], [3], [4,5,6]]` should return `[1,2,3,4,5,6]`.

```
fun flatten (nil) = nil
  | flatten (x::list) = x @ flatten(list);
```

7. Write a function `flatten2` that takes a list of 2-tuples and return a flat-tened list.
   - For example, `flatten2 [(1,2), (3,4), (5,6)]` should return `[1,2,3,4,5,6]`.

```
fun flatten2 (nil) = nil
  | flatten2 ((e1, e2)::list) = e1::e2::flatten2(list);
```