

CMSC 330: Homework 1 – Basic OCaml

Due Wednesday, October 15 **in section**

Implement the following functions in OCaml. Turn in a **printout** of your code to your TA. Do not use any functions in the `List` module, i.e., write every function from scratch using just list notation `[...]`, `::`, and pattern matching.

1. Write a function `prod l : int list -> int` that returns the product of the elements of `l`. The function `prod` should return 1 if the list is empty.
2. Write a function `addTail l e : 'a list -> 'a -> 'a list` that takes a list `l` and a single element `e` and returns a new list where `e` is appended to the *back* of `l`. For example, `addTail [1;2] 3 = [1;2;3]`.
3. Write a function `index l e : 'a list -> 'a -> int` that takes a list and a single element and returns the position of the last occurrence of that element in the list (indexed by zero). You should return -1 if the element is not in the list.
4. Write a function `unzip l : ('a*'b) list -> ('a list)*('b list)` that given a list of pairs, returns a pair of lists with the elements in the same order. For example, `unzip [(1, 2); (3, 4)] = ([1; 3], [2;4])`.
5. Write a function `app_int f m n : (int->'a)->int->int->'a list` that returns the list `[f m; f (m+1); ...; f n]`. It should return the empty list if `n < m`.