1. Define a function `findpos` which takes a list `ls` and a value `x` and returns its last (i.e., rightmost) position in the list: the output should be of type `int option`. If `x` does not occur in the list the output should be `NONE`.

(Note: Positions start from 0.)

Any other function you define should be `local` to the definition of `findpos`.

- `findpos`; val it = fn: `'a list * `'a -> int option
- `findpos([2,4,8,8,4], 2); val it = SOME 0: int option
- `findpos([2,4,8,8,4], 3); val it = NONE: int option
- `findpos([2,4,8,8,4], 4); val it = SOME 4: int option
- `findpos([2,4,8,8,4], 8); val it = SOME 3: int option
- `findpos(["its", "a", "mad", "mad", "world"], "mad"); val it = SOME 4: int option
- `findpos([[1,3], [2,3], [2,2]], [3,1]); val it = NONE: int option

2. Univariate polynomials with integer coefficients (i.e., polynomials over \( \mathbb{Z}[x] \)) can be represented by lists of integers. The idea is to represent \( a_0 + a_1 x + \ldots + a_n x^n \) by the list \[ [a_0, a_1, \ldots, a_n] \].

Write a `curried` function `eval` which takes such a list and an integer `c` as inputs and returns the value obtained by evaluating the polynomial at value `c`.

Any other function you define should be `local` to the definition of `eval`.

- `eval`; val it = fn: int list -> int -> int
- `eval [1,1,1] 2; val it = 7: int
- `eval [~1,3,5] 2; val it = 25: int