CSC 301 Lab 4
Baheem Ferrell

Tutorial two : Types

1. Predict the type of each of these objects. Check your answer by entering each followed by the semicolon.

   ("two" , true , 2);
   string * bool * int
   ["two", true , 2]; ?
   ((1,2) , (3,4,5));
   (int * int) * (int * int * int)
   [[1,2] , [3,4,5]];
   int list list
   [[ ] , [ ] , [ ]];
   'a list list
   ("andrew" , size "andrew");
   string * int
   (["andrew" = "andrew" , "andrew" = "ben"] , size "andrew" , size);
   bool list * int * int

2. Consider the following bindings. Try to predict the result before executing them.

   val (a , b) = (5 , 6);
   val a = 5: int; val b = 6: int;
   val (c , d) = (2 , ("xx" , “yy"));
   val c = 2: int; val d = ("xx", "yy"): string * string;
   val (e , (f , g)) = (1 , (2 , 3));
   val e = 1: int; val f = 2: int;
   val g = 3: int;
   val (l , m , n) = ("xx" , (1 , 2)); val l = “xx”: string; val m = (1,2): int * int; val n = ?;
   val (p, _) = (12, 10);
   val p = 12: int;
   val (q, 10) = (12, 10);
   val q = 12: int;
   val (r, 11) = (12, 10);
   Error because 11 != 10
   val u = 1::[2,3];
   val u = [1, 2, 3]: int list;

val v::w = 1::[2,3];
val v = 1: int; val w = [2, 3]: int list;
val h::t = [4,5,6];
val h = 4: int; val t = [5, 6]: int list;

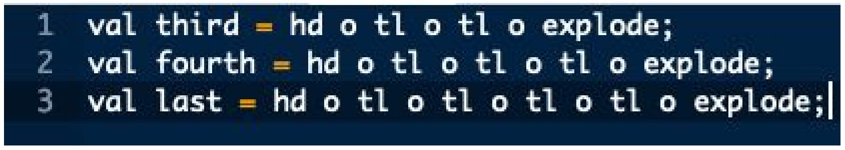
3. Consider the type of each of the following functions

   fun fone(x:int) = [x,x,x];
   int list
   fun ftwo(x) = (x,x,x);
   'a * 'a * 'a
   fun fthree(x,y) = [x ^ "b", y];
   string list
   fun ffour(x,y,z) = (x+(size y),z);

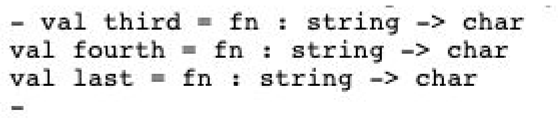
4. Find out the type of each of the in-built functions explode, rev, hd and tl. Try each function on a list or a string as appropriate. Make a note of each of the following: The type of explode and the name of its inverse. The type of rev and its inverse. The type of hd and what it is an abbreviation for The type of tl and what it is an abbreviation for. Evaluate each of the following; try to predict the result.

   hd(explode "south"); - type: char - s
   hd(tl(explode "north")); - type: char - o
   hd(rev(explode "east")); - type: char - t
   hd(tl(rev(explode "west"))); - type: char -s

5. The function composition operator is o (say of). We can use it to create functions from functions.



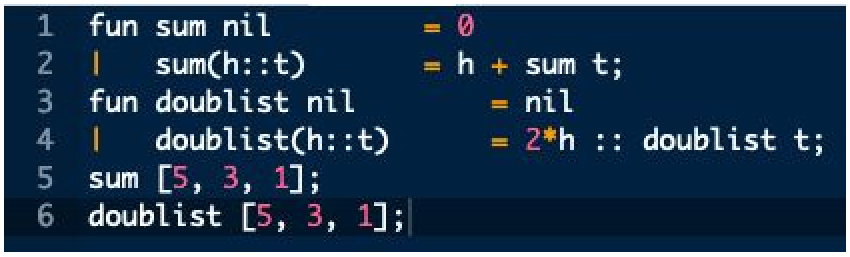
```
val third = hd o tl o tl o explode;
val fourth = hd o tl o tl o tl o explode;
val last = hd o tl o tl o tl o tl o explode;
```



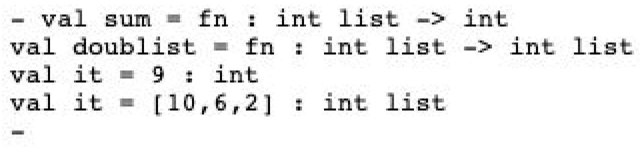
```
- val third = fn : string -> char
val fourth = fn : string -> char
val last = fn : string -> char
-
```

Recursion with lists

1. Define sum and doublist as shown. Execute both functions on the list [5,3,1].

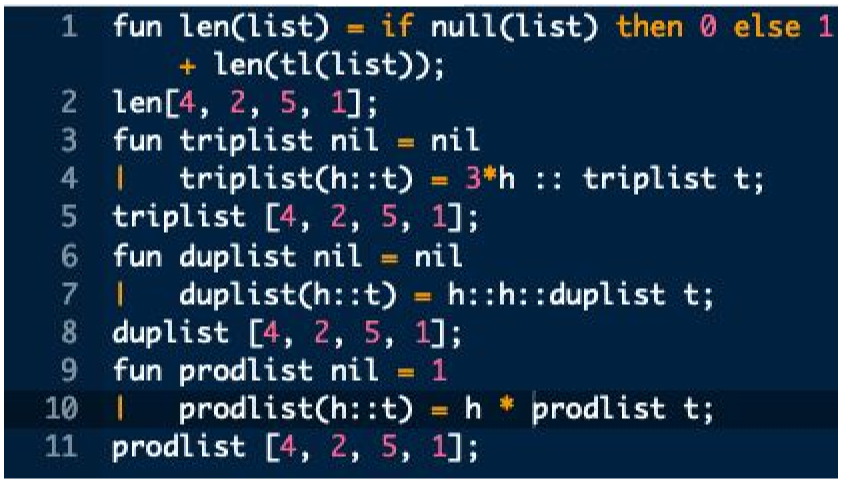


```
fun sum nil          = 0
|   sum(h::t)        = h + sum t;
fun doublist nil        = nil
|   doublist(h::t)      = 2*h :: doublist t;
sum [5, 3, 1];
doublist [5, 3, 1];
```

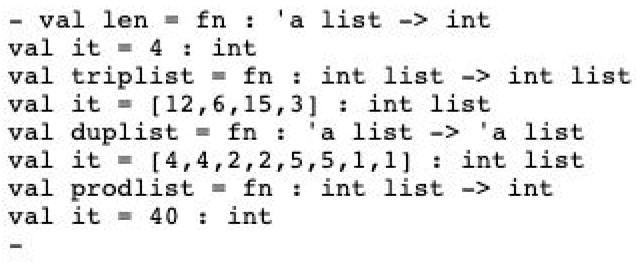


```
- val sum = fn : int list -> int
val doublist = fn : int list -> int list
val it = 9 : int
val it = [10,6,2] : int list
-
```

2. Define and test the following functions. An example execution has been given for each case.



```
fun len(list) = if null(list) then 0 else 1 + len(tl(list));
len[4, 2, 5, 1];
fun triplist nil = nil
|   triplist(h::t) = 3*h :: triplist t;
triplist [4, 2, 5, 1];
fun duplist nil = nil
|   duplist(h::t) = h::h::duplist t;
duplist [4, 2, 5, 1];
fun prodlist nil = 1
|   prodlist(h::t) = h * prodlist t;
prodlist [4, 2, 5, 1];
```

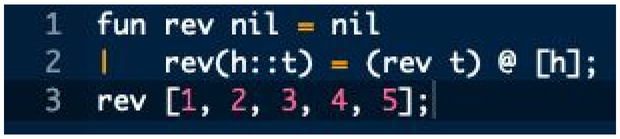


```
- val len = fn : 'a list -> int
val it = 4 : int
val triplist = fn : int list -> int list
val it = [12,6,15,3] : int list
val duplist = fn : 'a list -> 'a list
val it = [4,4,2,2,5,5,1,1] : int list
val prodlist = fn : int list -> int
val it = 40 : int
-
```

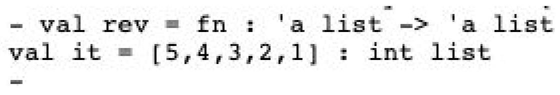
3. Define the function vallist which turns character digits into integers. You will need to use the in-built function ord.

    fun vallist nil = nil
    | vallist(h::t) = (ord h - ord "0") :: vallist t;

4. Define the function rev which reverses the order of a list. Note the comments made about adding elements to lists earlier.

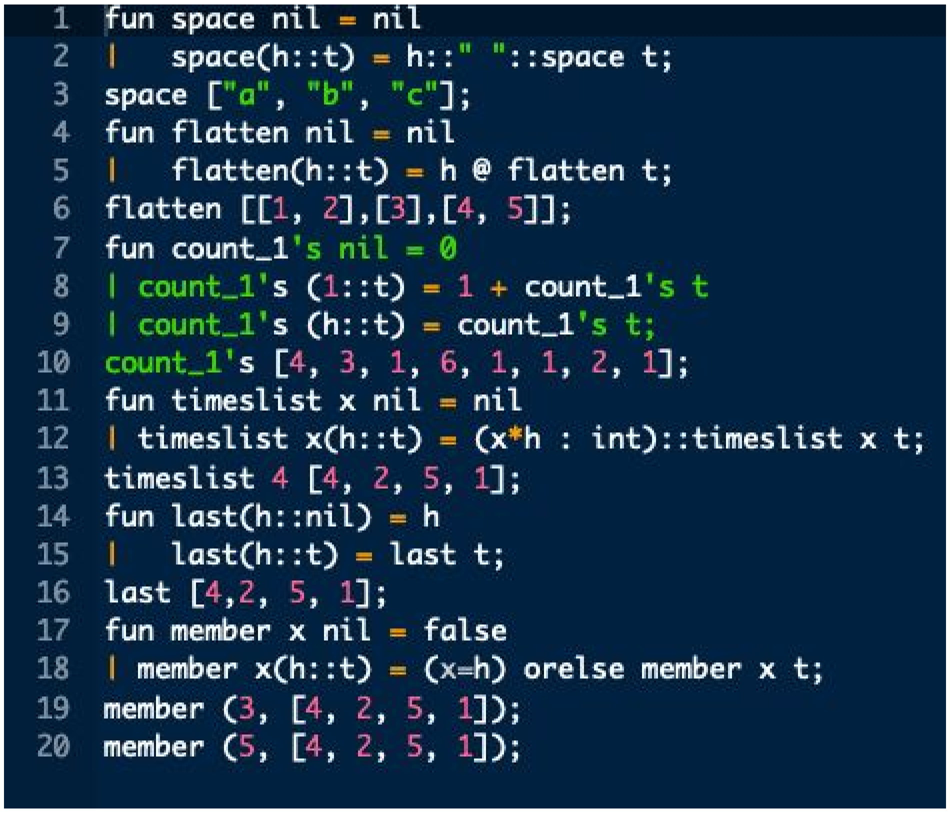


```
fun rev nil = nil
|   rev(h::t) = (rev t) @ [h];
rev [1, 2, 3, 4, 5];
```



```
- val rev = fn : 'a list -> 'a list
val it = [5,4,3,2,1] : int list
-
```

5. Define the following functions.



```
fun space nil = nil
|   space(h::t) = h::" "::space t;
space ["a", "b", "c"];
fun flatten nil = nil
|   flatten(h::t) = h @ flatten t;
flatten [[1, 2],[3],[4, 5]];
fun count_1's nil = 0
| count_1's (1::t) = 1 + count_1's t
| count_1's (h::t) = count_1's t;
count_1's [4, 3, 1, 6, 1, 1, 2, 1];
fun timeslist x nil = nil
| timeslist x(h::t) = (x*h : int)::timeslist x t;
timeslist 4 [4, 2, 5, 1];
fun last(h::nil) = h
|   last(h::t) = last t;
last [4,2, 5, 1];
fun member x nil = false
| member x(h::t) = (x=h) orelse member x t;
member (3, [4, 2, 5, 1]);
member (5, [4, 2, 5, 1]);
```

```
- val space = fn : string list -> string list
val it = ["a"," ","b"," ","c"," "] : string list
val flatten = fn : 'a list list -> 'a list
val it = [1,2,3,4,5] : int list
val count_1's = fn : int list -> int
val it = 4 : int
val timeslist = fn : int -> int list -> int list
val it = [16,8,20,4] : int list
stdIn:14.5-15.24 Warning: match nonexhaustive
          h :: nil => ...
          h :: t => ...

val last = fn : 'a list -> 'a
val it = 1 : int
stdIn:18.22 Warning: calling polyEqual
val member = fn : ''a -> ''a list -> bool
val it = fn : (int * int list) list -> bool
val it = fn : (int * int list) list -> bool
-
```