1	Erre a feladatlapra **ne írjon semmit!** Használja a *Könyvtári SML-függvények típusa* c. összeállítást (a feladatlap hátoldalán található). Az E(lméleti) és a Gy(akorlati) feladat 7-7 pontot, a P(rogramozás) feladat 21 pontot ér.

E. Adjon rövid összefoglalót az SML-nyelv egyszerű és összetett adattípusairól! Részletesen mutassa be az összetett adattípusok közül az ennest és a rekordot. Mi a kapcsolat az ennes és a rekord között? Mutassa be, hogyan lehet elérni egy ennes, ill. egy rekord összetevőit.

Gy. 1. és 2. részfeladat: mi az `x` **értéke** a deklaráció kiértékelése után? 3. részfeladat: a kifejezésben két hiba van, melyek ezek? (A szintaktikailag helyes kifejezés, ill. deklarációk függetlenek egymástól, a szükséges könyvtárak be vannak töltve.)

1. `val x = ((Math.cos Math.pi, 3), #1 (2, 2.0), chr(3 + ord #"a"))`
2. `val x = map (String.fields Char.isAlpha) ["4aa55bb666", "56", "b"]`
3. `let val (x, y, z) = (#"x", "y", 5, 4.0) in [ord x + ord y, z] end`

P. Először a **segédfüggvényt** írja meg, majd a *felhasználásával* oldja meg a **teljes feladatot!** Tartsa be mindkét specifikációt!

Segédfüggvény írása: írjon két függvényt `vanL`, ill. `dbE` néven, amelyek egy

```
datatype 'a fa = E | L of 'a | N of 'a fa * 'a fa * 'a fa
```

deklarációval megadott **fákból képezett listákon** működnek. `vanL` megvizsgálja, hogy van-e kizárólag `L` levélből álló fa a listában: ha van, 1-gyel tér vissza, ha nincs, 0-val. `dbE` megszámolja, hogy hány darab `E` levélből álló fa van a listában.

```
vanL fs = 1, ha az fs fa listában van L _, 0 különben
dbE fs = hány darab E van az fs fa listában
vanL : 'a fa list -> int
dbE : 'a fa list -> int

Példák: vanL []                    = 0
        vanL [E, L 1, N(E,E,E)]    = 1
        vanL [L 2, N(L 3,L 4,L 5)] = 1

        dbE  []                   = 0
        dbE  [E, N(E,L #"a",E)]   = 1
        dbE  [E, N(L 5,E,L 2), E] = 2
```

Teljes feladat: Írjon függvényt `lTestverE` néven a `vanL` és a `dbE` függvények felhasználásával, amely egy fenti deklarációval megadott fában meghatározza azoknak az `E` leveleknek a számát, amelyeknek legalább egy `L` testvérük van! Testvéreknek azokat a leveleket (`L x` vagy `E`) nevezzük, amelyek közös `N` csomóponton lógnak. További **segédfüggvényeket ne definiáljon!**

```
lTestverE f = azoknak az E-eknek a száma az f fában, amelyeknek legalább egy L
    testvérük van
lTestverE : 'a fa -> int

Példák: lTestverE E                                  = 0
        lTestverE (L 3)                              = 0
        lTestverE (N(E,E,E))                         = 0
        lTestverE (N(L 9, E, N(L 5,E,E)))            = 3
        lTestverE (N(L 7, N(L 8, N(L 9,E,E), E), E)) = 4
```

Könyvtári SML-függvények típusa

Név	*Könyvtár*	*Típus*
::	List	'a * 'a list -> 'a list
@	List	'a list * 'a list -> 'a list
^	String	string -> string -> string
all	List	('a -> bool) -> 'a list -> bool
app	List	('a -> unit) -> 'a list -> unit
before	General	'a * 'b -> 'a
ceil	General	real -> int
chr	Char	int -> char
compare	Char	char * char -> order
compare	Int	int * int -> order
compare	Real	real * real -> order
compare	String	string * string -> order
concat	List	'a list list -> 'a list
cos	Math	real -> real
div	Int	int * int -> int
drop	List	'a list * int -> 'a list
exists	List	('a -> bool) -> 'a list -> bool
exp	Math	real -> real
explode	String	string -> char list
fields	String	(char -> bool) -> string -> string list
filter	List	('a -> bool) -> 'a list -> 'a list
find	List	('a -> bool) -> 'a list -> 'a option
floor	General	real -> int
foldl	List	('a * 'b -> 'b) -> 'b -> 'a list -> 'b
foldr	List	('a * 'b -> 'b) -> 'b -> 'a list -> 'b
fromString	Int	string -> int option
fromString	Real	string -> real option
hd	List	'a list -> 'a
implode	String	char list -> string
isAlpha	Char	char -> bool
isAlphaNum	Char	char -> bool
isDigit	Char	char -> bool
isHexDigit	Char	char -> bool
isLower	Char	char -> bool
isPrefix	String	string -> string -> bool
isPunct	Char	char -> bool
isSome	Option	'a option -> bool

Név	*Könyvtár*	*Típus*
isSpace	Char	char -> bool
isUpper	Char	char -> bool
last	List	'a list -> 'a
length	List	'a list -> int
ln	Math	real -> real
map	List	('a -> 'b) -> 'a list -> 'b list
mapPartial	List	('a -> 'b option) -> 'a list -> 'b list
mod	Int	int * int -> int
nth	List	'a list * int -> 'a
o	General	('a -> 'b) * ('c -> 'a) -> 'c -> 'b
ord	Char	char -> int
partition	List	('a -> bool) -> 'a list -> 'a list * 'a list
pi	Math	real
pow	Math	real * real -> real
print	TextIO	string -> unit
printVal	Meta	'a -> 'a
quot	Int	int * int -> int
real	General	int -> real
rem	Int	int * int -> int
rev	List	'a list -> 'a list
revAppend	List	'a list * 'a list -> 'a list
round	General	real -> int
sin	Math	real -> real
size	String	string -> int
sqrt	Math	real -> real
str	String	char -> string
sub	String	string * int -> char
take	List	'a list * int -> 'a list
tl	List	'a list -> 'a list
toLower	Char	char -> char
toString	Int	int -> string
toString	Real	real -> string
toUpper	Char	char -> char
tokens	String	(char -> bool) -> string -> string list
trunc	General	real -> int
valOf	Option	'a option -> 'a

7-bites ASCII-kódú karakterek sorrendje (`chr 32`-től `chr 126`-ig)

```
 32 - 63    ␣ ! " # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
 64 - 95   @ A B C D E F G H I J K L M N O P Q R S T U V W X Y Z [ \ ] ^ _
 96 - 126  ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~
```