# Webes és mobil programozás F#-ban 2. gyakorlat

## Berezvai Dániel jegyzete <http://elte.3ice.hu/>

Óra fele elment a CloudSharper telepítéssel.

De legalább szereztem egy admin share útvonalat:)

Loot [\\nas2\needed\](file:///%5C%5Cnas2%5Cneeded%5C) from the school machines.

Kezdődik az óra…

→ ELTE\F#\2\Lesson2.fs.md

## Előző HF megoldása

let fun1 (x::xs) = (List.length x, x);;

### Fut

> fun1 [[1;2;3];[4;5;6];[7;8;9]];;

val it : int \* int list = (3, [1; 2; 3])

## Másik HF

let d x =

 match x with

 | \_ when x%2=1 -> 1

 | \_ -> x/2;;

Szerintem az enyém jobb (→ 1.docx)

## Option típus

Some – valami

None – Valami nem sikerült.

**let** optionFunction **(**list**,**a**)** **=**

 **match** list **with**

 **|** **[]** **->** None

 **|** **\_** **->** Some a

**let** optionResult **=** optionFunction **([**1.**.**5**],** List**.**length **[**1.**.**5**])**

## Feladat: Függvény, ami párt vár. Ha az első eleme páratlan, akkor visszaadja a második elemét, ha nem, akkor pedig None-t ad vissza

**let** optionFunction **(**a**,**b**)** **=**

 **match** a**%**2 **with**

 **|** 0 **->** None

 **|** **\_** **->** Some b**;;**

**>** optionFunction **(**1**,**2**);;**

**val** it **:** int option **=** Some 2

**>** optionFunction **(**2**,**1**);;**

**val** it **:** int option **=** None

Még fut:

**>** optionFunction **(**3**,** **[**1.**.**5**]);;**

**val** it **:** int list option **=** Some **[**1**;** 2**;** 3**;** 4**;** 5**]**

**>** optionFunction **(**2**,** **[**1.**.**5**]);;**

**val** it **:** int list option **=** None

## Fordítva és másképp

**let** optionFunction **(**a**,**b**)** **=**

 **match** a **with**

 **|** **\_** **when** a **%** 2 **=** 0 **->** Some b

 **|** **\_** **->** None**;;**

## Feladat: Listát várunk. Ha a kapott lista vége (xs) hosszabb, mint 5, akkor a kapott lista végével térünk vissza, különben None-nal.

Néha eltértünk az 5-től, figyelembe kellene venni, hogy $xs$ vagy $l$ hosszát nézzük.

**let** long **(**x**::**xs**)** **=**

 **match** xs **with**

 **|** **\_** **when** List**.**length xs **>** 5 **->** Some xs

 **|** **\_** **->** None

Ez figyelmeztetést ad (Üres lista meg is öli…)

Másképp:

**let** long l **=**

 **match** l **with**

 **|** **\_** **when** List**.**length l **>** 5 **->** Some **(**List**.**tail l**)**

 **|** **\_** **->** None

Üres listát is kezelve:

**let** long l **=**

 **match** l **with**

 **|** **[]** **->** None

 **|** x**::**xs **when** List**.**length xs **>** 5 **->** Some xs

 **|** **\_** **->** None**;;**

(Szerintem ez még mindig hibát ad üres listára.)

## Rekurzió (Lista elemenkénti feldolgozására is jó)

Imperatív nyelvek ciklikuságát helyettesíti.

### Faktoriális

**let** **rec** factorial n **=**

 **match** n **with**

 **|** 1 **->** 1

 **|** **\_** **when** n **>** 1 **->** n **\*** factorial **(**n**-**1**)**

 **|** **\_** **->** 0

Vegyük észre a $rec$ kulcsszót. Van legalább egy sor, ahol megáll. (Most 2 van.) És legalább egy sora, ahol önmagát meghívja. (Most pont 1 van.)

Érdemes a rekurzív hívást a sor végére hagyni. Memóriát nem eszi annyira.

Mintamegadás sorrendje számít. Legfelül legyenek a megállás feltételek, legalul a rekurzív hívások.

## Feladat: $Sum$

**let** **rec** sum l **=**

 **match** l **with**

 **|** **[]** **->** 0

 **|** **(**x**::**xs**)** **->** x **+** sum xs**;;**

**val** sum **:** int list **->** int

**>** sum **[**1.**.**5**];;**

**val** it **:** int **=** 15

**>** sum **[**1.**.**100**];;**

**val** it **:** int **=** 5050

**>** sum **[**1.**.**2**];;**

**val** it **:** int **=** 3

## Feladat: $isPrefixOf$

Két stringet hasonlítunk egymással. Karakterenként le tudjuk kérni az elemeket.

Túl nehéz, majd együtt megoldjuk óra végén.

UPDATE: Nem került rá sor.

## Feladat: $elem$

Listában elem benne van-e.

**let** **rec** elem e list **=**

 **match** list **with**

 **|** **[]** **->** **false**

 **|** x**::**xs **when** e **=** x **->** **true**

 **|** x**::**xs **->** elem e xs**;;**

**val** elem **:** e**:**'a **->** list**:**'a list **->** bool **when** 'a **:** equality

**>** elem 5 **[**1.**.**5**];;**

**val** it **:** bool **=** **true**

**>** elem 5 **[**1.**.**4**];;**

**val** it **:** bool **=** **false**

## Feladat: $take$

**let** **rec** take n l **=**

 **match** l **with**

 **|** **[]** **->** **[]**

 **|** x**::**xs **when** n**>**0 **->** x **::** take **(**n**-**1**)** xs

 **|** **\_** **->** **[];;**

**val** take **:** n**:**int **->** l**:**'a list **->** 'a list

**>** take 5 **[**1.**.**5**];;**

**val** it **:** int list **=** **[**1**;** 2**;** 3**;** 4**;** 5**]**

**>** take 2 **[**1.**.**5**];;**

**val** it **:** int list **=** **[**1**;** 2**]**

**>** take 6 **[**1.**.**5**];;**

**val** it **:** int list **=** **[**1**;** 2**;** 3**;** 4**;** 5**]**

## Feladat: $drop$, stb

Szorgalmi, gyakorlás.

Gyakorlat vége.