February 13, 2020

If The code is in red, it's a failed attempt!!!!!!!!!!!!!!!!!!

These notes are not endorsed by anyone, not even me. Proceed at your own risk.

Last night I put up the midterm of last time last year.

We'll go through a bunch wrong solutions together and find the errors.

higher-order function

(f,n) -> $\underbrace{f \ldots f}_{\text{n-times}}$

```
let rec repeated (f,n) = ...<code removed> ...
        'a -> 'a * int -> 'a -> 'a
```

The following: gets 0%. The wrong type and a wrong base case.

```
= if n=0 then f
        else f(f, n-1)
```

let's try again

The following doesn't type check 45%

```
=let  if n=0 then f
        else f(repeated(f,n-1))
```

The following is better but still wrong

The only thing wrong is the base case 66%

```
if n=0 then f
else
        fun x -> repeated(f,n-1)(f x)
```

```
if n=0 then fun x -> x
else fun x -> repeated (f, n-1)(f x)
```

Arrays as lists of lists

m1 = [[1;2;3];[4;5;6];[7;8;9]]

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

proper if all rows have the same length

square if it is proper & the same length of each row = number of rows

```
let square m =
        match m with
        |[] -> true
        |_ -> List.for_all ( fun r -> (List.length m) = (List.length r) ) m
```

Write down the list. ___ on your cheat sheet.

Now let me lead you through the saga of the tortured mind

```
let square2 m =
        let l = List.length m in
        match m with
        |[] -> true
        | first::rest -> let rec helper mat =
                if(List.length first) = m then
                then helper rest else false
```

```
let square2 m=
        let l = List.length m in
        match m with
        | [] -> true
        | first :: rest -> let rec helper mat =
        ...
        in
        helper m;;
```

This next example is a mistake I did last night

```
let List.length m = l in
```

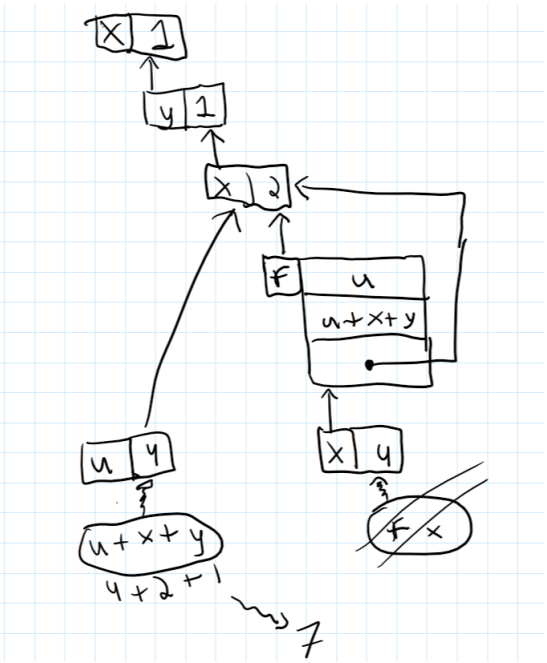
```
let l = List.length m in
let rec helper m =
        match mat with
        |[] -> true
        | first :: rest -> if(List.length first) = l
                then helper rest else false
```

```
let l = List.length m in
let rec helper mat =
        match mat with
        |[] -> true
        | first :: rest -> if(List.length first) = l
                then helper rest else false
```

```
let proper2 m =
        match m with
        |[] -> true
        | first :: rest -> let l = List.length first in
        ...
```

Make sure you can do all homework questions involving recursions on lists.

```
let x=1 in
        let y=x in
                let x=2 in
                        let f u = u + x + y in
                                let x = 4 in
                                        f x
```



Following is a question by a student

```
let x = 2 in
        let f = let c = 3 in
                fun u -> u+1
        in
                f x
```