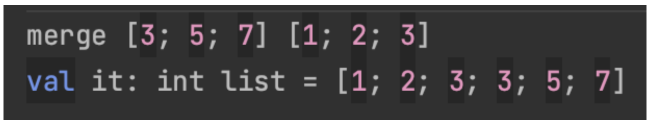
Programmer som Data - Assignment 5

Bastjan Rosgaard Sejberg, Søren Kastrup, Weihao Chen Nyholm-Andersen

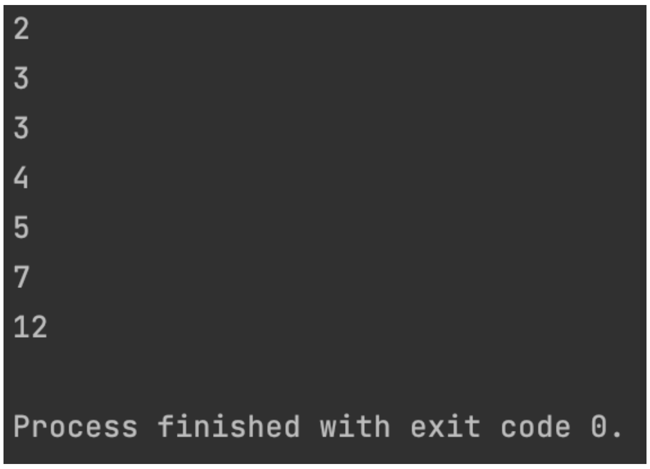
October 2023

5.1



```
merge [3; 5; 7] [1; 2; 3]
val it: int list = [1; 2; 3; 3; 5; 7]
```

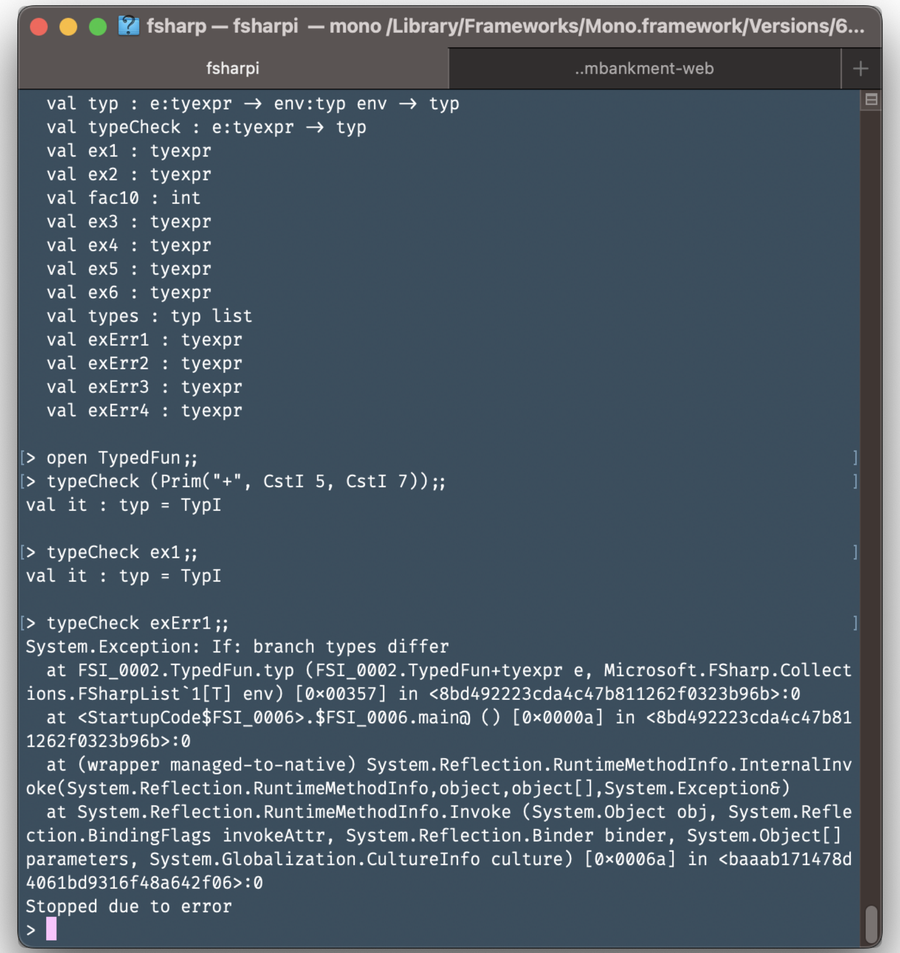
*For exercise 5.1a refer to the files in the folder **Exercise_5.1-5.7**.*



```
2
3
3
4
5
7
12

Process finished with exit code 0.
```

*For exercise 5.1b refer to the files in the folder **Exercise_5.1_cs**.*

5.7

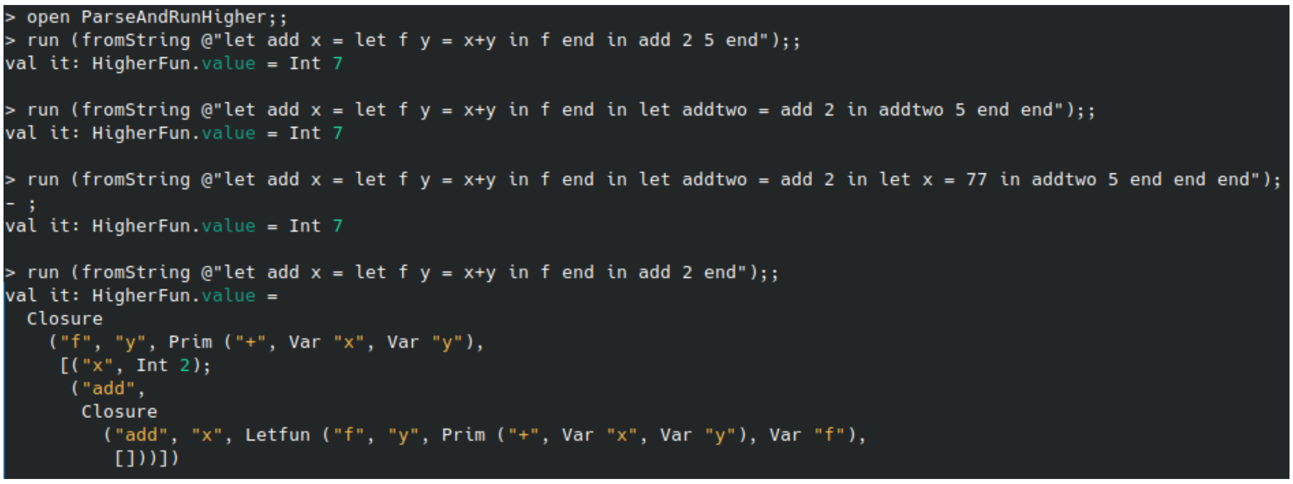


fsharp — fsharpi — mono /Library/Frameworks/Mono.framework/Versions/6...

```
  val typ : e:tyexpr → env:typ env → typ
  val typeCheck : e:tyexpr → typ
  val ex1 : tyexpr
  val ex2 : tyexpr
  val fac10 : int
  val ex3 : tyexpr
  val ex4 : tyexpr
  val ex5 : tyexpr
  val ex6 : tyexpr
  val types : typ list
  val exErr1 : tyexpr
  val exErr2 : tyexpr
  val exErr3 : tyexpr
  val exErr4 : tyexpr

> open TypedFun;;
> typeCheck (Prim("+", CstI 5, CstI 7));;
val it : typ = TypI

> typeCheck ex1;;
val it : typ = TypI

> typeCheck exErr1;;
System.Exception: If: branch types differ
  at FSI_0002.TypedFun.typ (FSI_0002.TypedFun+tyexpr e, Microsoft.FSharp.Collect
ions.FSharpList`1[T] env) [0x00357] in <8bd492223cda4c47b811262f0323b96b>:0
  at <StartupCode$FSI_0006>.$FSI_0006.main@ () [0x0000a] in <8bd492223cda4c47b81
1262f0323b96b>:0
  at (wrapper managed-to-native) System.Reflection.RuntimeMethodInfo.InternalInv
oke(System.Reflection.RuntimeMethodInfo,object,object[],System.Exception&)
  at System.Reflection.RuntimeMethodInfo.Invoke (System.Object obj, System.Refle
ction.BindingFlags invokeAttr, System.Reflection.Binder binder, System.Object[]
parameters, System.Globalization.CultureInfo culture) [0x0006a] in <baaab171478d
4061bd9316f48a642f06>:0
Stopped due to error
>
```

*Refer to the files in the folder **Exercise_5.1-5.7**.*

6.1

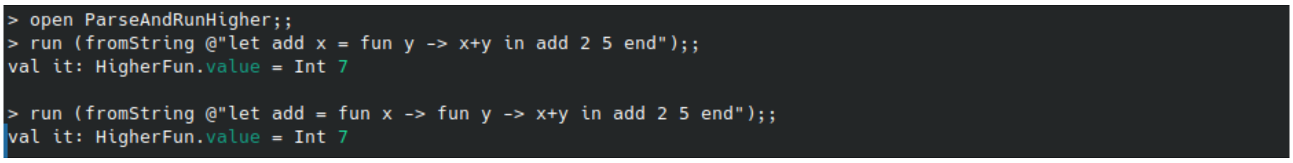


```
> open ParseAndRunHigher;;
> run (fromString @"let add x = let f y = x+y in f end in add 2 5 end");;
val it: HigherFun.value = Int 7

> run (fromString @"let add x = let f y = x+y in f end in let addtwo = add 2 in addtwo 5 end end");;
val it: HigherFun.value = Int 7

> run (fromString @"let add x = let f y = x+y in f end in let addtwo = add 2 in let x = 77 in addtwo 5 end end end");
- ;
val it: HigherFun.value = Int 7

> run (fromString @"let add x = let f y = x+y in f end in add 2 end");;
val it: HigherFun.value =
  Closure
    ("f", "y", Prim ("+", Var "x", Var "y"),
     [("x", Int 2);
      ("add",
       Closure
         ("add", "x", Letfun ("f", "y", Prim ("+", Var "x", Var "y"), Var "f"),
          []))])
```

The result of the third program is as expected, since the variable x is set to a value of 77, however the variable itself is never used with $addTwo\ 5 -> add\ 2\ 5 -> 7$ being the final calculation.

The last program never returns an actual result from the calculation and instead returns the code parsed from the inputted string, which is the case since the method *add* requires two numbers as arguments to be able to return a result but only one is ever provided.

6.2 & 6.3

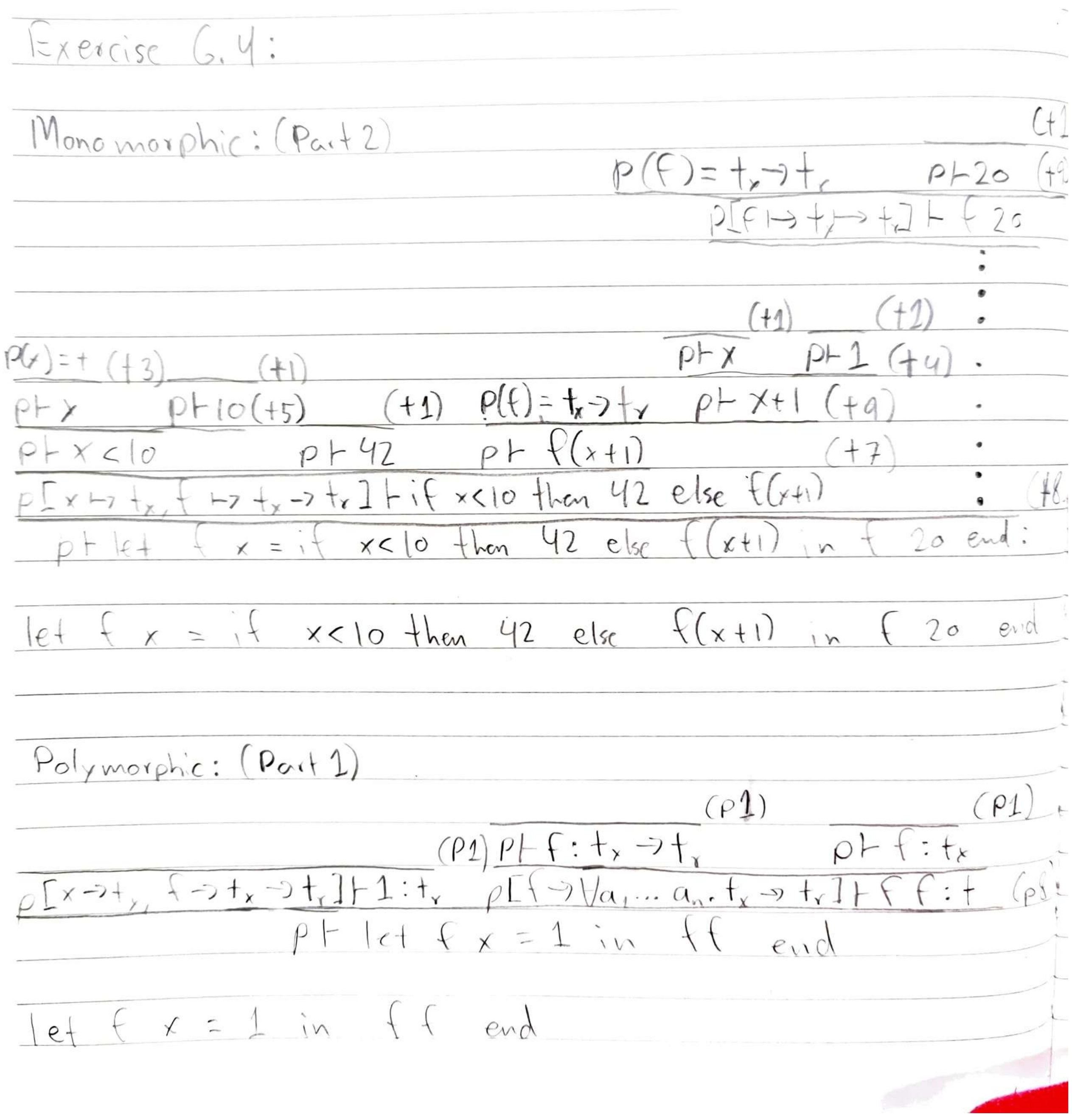


```
> open ParseAndRunHigher;;
> run (fromString @"let add x = fun y -> x+y in add 2 5 end");;
val it: HigherFun.value = Int 7

> run (fromString @"let add = fun x -> fun y -> x+y in add 2 5 end");;
val it: HigherFun.value = Int 7
```

For exercise 6.2 refer to the files ***Absyn.fs*** *and* ***HigherFun.fs*** *in the folder* ***Exercise_6.2-6.3****.*

For exercise 6.3 refer to the files ***FunLex.fsl*** *and* ***FunPar.fsy*** *in the folder* ***Exercise_6.2-6.3****.*

6.4



Exercise 6.4:

Monomorphic: (Part 2)

$$\dfrac{\dfrac{\rho(f) = t_x \to t_r \qquad \rho \vdash 20}{\rho[f \mapsto t_x \to t_r] \vdash f\ 20}}{\vdots}\ (t1\ (t9$$

$$\dfrac{\dfrac{\dfrac{\rho(x) = t\ (t3)}{\rho \vdash x} \quad \dfrac{}{\rho \vdash 10}\ (t5)}{\rho \vdash x < 10}\ (t1) \quad \dfrac{}{\rho \vdash 42}\ (t1) \quad \dfrac{\rho(f) = t_x \to t_r \quad \dfrac{\dfrac{}{\rho \vdash x}\ (t1) \quad \dfrac{}{\rho \vdash 1}\ (t2)}{\rho \vdash x+1}\ (t4)}{\rho \vdash f(x+1)}\ (t9)}{\dfrac{\rho[x \mapsto t_x, f \mapsto t_x \to t_r] \vdash \text{if } x<10 \text{ then } 42 \text{ else } f(x+1)\ (t7)}{\rho \vdash \text{let } f\ x = \text{if } x<10 \text{ then } 42 \text{ else } f(x+1) \text{ in } f\ 20 \text{ end}:}}\ (t8$$

let f x = if x<10 then 42 else f(x+1) in f 20 end

Polymorphic: (Part 1)

$$\dfrac{\rho[x \to t_x, f \to t_x \to t_r] \vdash 1 : t_r \ (P1) \quad \dfrac{\dfrac{}{\rho \vdash f : t_x \to t_r}\ (P1) \quad \dfrac{}{\rho \vdash f : t_x}\ (P1)}{\rho[f \to \forall \alpha_1 \ldots \alpha_n . t_x \to t_r] \vdash f\ f : t}\ (P5}{\rho \vdash \text{let } f\ x = 1 \text{ in } f\ f \text{ end}}$$

let f x = 1 in f f end

Polymorphic is when a function is able to operate with different types, so in this case the function may have either the type int or float.

Monomorphic (part 2) is when the function type is set in stone or is fixed. So in this case the type for this function is highly likely integer.

The biggest difference is that with Polymorphic, the function can EITHER be integer or float, while with Monomorphic the function can ONLY be an integer or float.

6.5

Part 1

- val it: string = ”int”
- Not Typable - This is not typable since it would go into a circularity meaning that it may end up in a infinite typing.
- val it: string = ”bool”
- Not Typable - Simply incorrect typing. Since bool is not int.
- val it: string = ”bool”

Part 2

- $let\ f\ x\ =\ true\ in\ f\ true\ end\ (bool\ \rightarrow\ bool)$
- $let\ f\ x\ =\ 2\ in\ f\ 42\ end\ (int\ \rightarrow\ int)$
- $let\ f\ x\ =\ let\ g\ y\ 3\ in\ g\ 6\ in\ f\ 42\ end\ (int\ \rightarrow\ int\ \rightarrow\ int)$
- $let\ f\ x\ =\ let\ g\ y\ =\ x\ in\ g\ end\ in\ f\ end\ ('a\ \rightarrow\ 'b\ \rightarrow\ 'a)$
- $let\ f\ x\ =\ let\ g\ y\ =\ y\ in\ g\ end\ in\ f\ end\ ('a\ \rightarrow\ 'b\ \rightarrow\ 'b)$
- $let\ f\ a\ =\ let\ g\ b\ =\ a\ let\ h\ c\ =\ c\ b\ in\ h\ end\ in\ g\ end\ in\ f\ end\ (('a\ \rightarrow\ 'b)\ \rightarrow\ ('b\ \rightarrow\ 'c)\ \rightarrow\ ('a\ \rightarrow\ 'c))$
- $let\ f\ a\ =\ f\ a\ in\ f\ end\ ('a\ \rightarrow\ 'b)$
- $let\ f\ x\ =\ f\ x\ in\ f\ 1\ end\ ('a)$