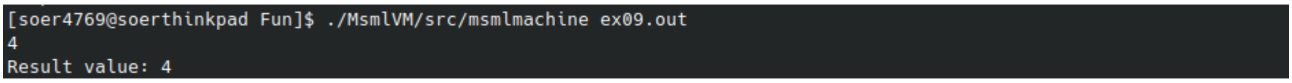
Programmer som Data - Assignment 12

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

December 2023

13.1

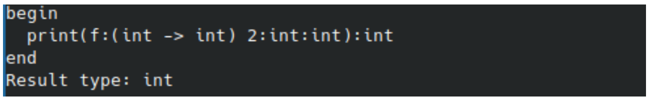
0.1 What is the result value of running ex09.out?



```
[soer4769@soerthinkpad Fun]$ ./MsmlVM/src/msmlmachine ex09.out
4
Result value: 4
```

The result value is 4.

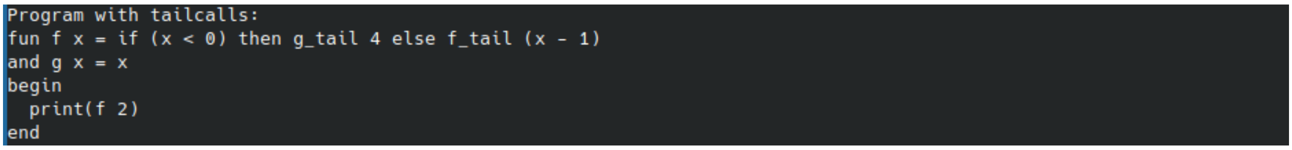
0.2 What type does the result value have?



```
begin
  print(f:(int -> int) 2:int:int):int
end
Result type: int
```

The type of the result value is integer.

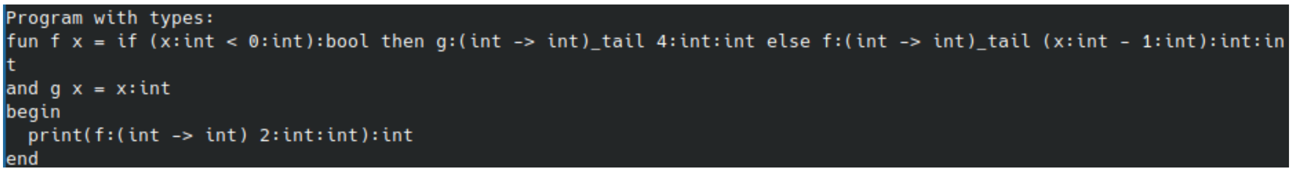
0.3 What application calls have been annotated as tail calls? Explain how this matches the intuition behind a tail call.



```
Program with tailcalls:
fun f x = if (x < 0) then g_tail 4 else f_tail (x - 1)
and g x = x
begin
  print(f 2)
end
```

Inside function *f*, the call to function *g* *(g_tail)* and *f* *(f_tail)* has been annotated as tail calls. This is because a tail call is recognised as a call to a function as the last action before a function returns hereby the naming scheme, thus since the last action of function *f* depends on the Boolean value of its if-statement the last action could thereby be either a call to function *g* or itself *f*.

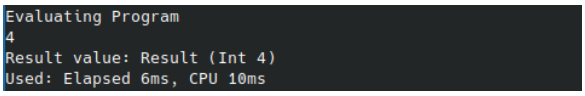
0.4 What type has been annotated for the call sites to the functions f and g? Function f is called in two places, and g in one place.



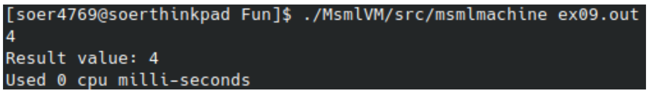
```
Program with types:
fun f x = if (x:int < 0:int):bool then g:(int -> int)_tail 4:int:int else f:(int -> int)_tail (x:int - 1:int):int:in
t
and g x = x:int
begin
  print(f:(int -> int) 2:int:int):int
end
```

Function *f:(int → int)* and *g:(int → int)*, has both been annotated as taking in an int and returning an int; *f* is called in the print statement and as a tail call within itself and *g* is called within function *f* as a tail call likewise.

0.5 What is the running time for executing the example using the evaluator, and what is the running time using the byte code ex09.out using msmlmachine?



```
Evaluating Program
4
Result value: Result (Int 4)
Used: Elapsed 6ms, CPU 10ms
```



```
[soer4769@soerthinkpad Fun]$ ./MsmlVM/src/msmlmachine ex09.out
4
Result value: 4
Used 0 cpu milli-seconds
```

The running time using *the evaluator* is 10 cpu-milliseconds, where for the *msmlmachine* it is less than 0 cpu-milliseconds.

0.6 Now compile the example ex09.sml without optimizations. How many byte code instructions did the optimization save for this small example?

```
...
LABEL LabFunc_f_L4
    38: GETBP
    39: CSTI 1
    41: ADD
    42: LDI
    43: CSTI 0
    45: LT
    46: IFZERO L7
    48: CSTI 2
    50: LDI
    51: CSTI 4
    53: CLOSCALL 1
    55: GOTO L6
LABEL L7
    57: GETBP
    58: CSTI 0
    60: ADD
    61: LDI
    62: GETBP
    63: CSTI 1
    65: ADD
    66: LDI
    67: CSTI 1
    69: SUB
    70: CLOSCALL 1
LABEL L6
    72: RET 2
...
```

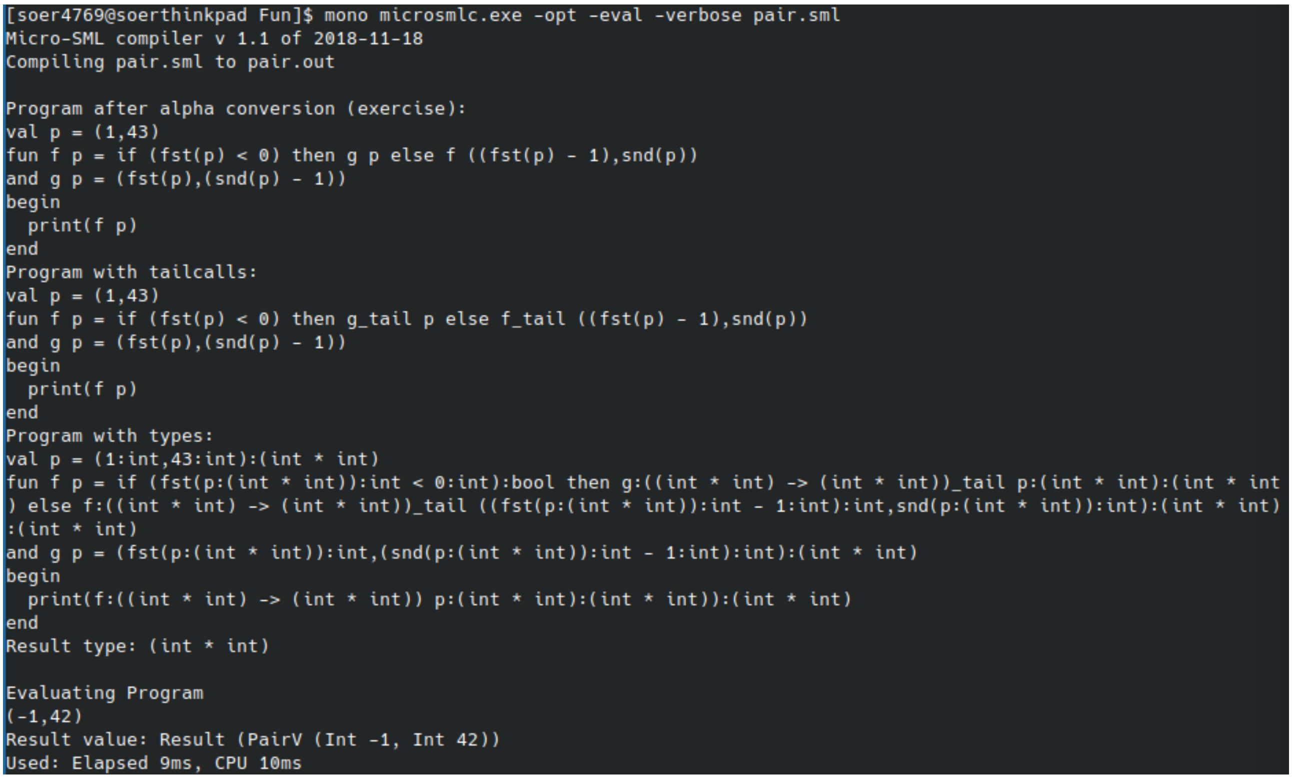
Listing 1: ex09.out (without opt.)

```
...
LABEL LabFunc_f_L4
    38: GETBP
    39: CSTI 1
    41: ADD
    42: LDI
    43: CSTI 0
    45: LT
    46: IFZERO L6
    48: CSTI 2
    50: LDI
    51: CSTI 4
    53: TCLOSCALL 1

LABEL L6
    55: GETBP
    56: LDI
    57: GETBP
    58: CSTI 1
    60: ADD
    61: LDI
    62: CSTI 1
    64: SUB
    65: TCLOSCALL 1




...
```

Listing 2: ex09.out (with opt.)

By comparing the output of byte code instructions in the console for both programs, it can be seen that the following 5 byte code instructions has been saved by optimisation: *55: GOTO L6*, *58: CSTI 0*, *60: ADD*, *LABEL L6*, *72: RET 2*. This can be seen in the sections where changes has occurred between the output with and without the optimisation flag enabled.

13.2



```
[soer4769@soerthinkpad Fun]$ mono microsmlc.exe -opt -eval -verbose pair.sml
Micro-SML compiler v 1.1 of 2018-11-18
Compiling pair.sml to pair.out

Program after alpha conversion (exercise):
val p = (1,43)
fun f p = if (fst(p) < 0) then g p else f ((fst(p) - 1),snd(p))
and g p = (fst(p),(snd(p) - 1))
begin
  print(f p)
end
Program with tailcalls:
val p = (1,43)
fun f p = if (fst(p) < 0) then g_tail p else f_tail ((fst(p) - 1),snd(p))
and g p = (fst(p),(snd(p) - 1))
begin
  print(f p)
end
Program with types:
val p = (1:int,43:int):(int * int)
fun f p = if (fst(p:(int * int)):int < 0:int):bool then g:((int * int) -> (int * int))_tail p:(int * int):(int * int
) else f:((int * int) -> (int * int))_tail ((fst(p:(int * int)):int - 1:int):int,snd(p:(int * int)):int):(int * int)
:(int * int)
and g p = (fst(p:(int * int)):int,(snd(p:(int * int)):int - 1:int):int):(int * int)
begin
  print(f:((int * int) -> (int * int)) p:(int * int):(int * int)):(int * int)
end
Result type: (int * int)

Evaluating Program
(-1,42)
Result value: Result (PairV (Int -1, Int 42))
Used: Elapsed 9ms, CPU 10ms
```

Refer to the files in the folder ***Exercise-13.2****. We could sadly not get the two questions regarding the type rules for the primitives fst and snd to work nor how to write evaluation rules for them. There is also a bug in TypeInference.fs for these primitives so that pair has to take integer types. Lastly there is a bug in the msml-machine.c which means when pair.out is run with it when compiled, it crashes with a segmentation dump error.*

13.3

Code:

```
begin
  let
    val y1 = 1
    fun f x =
      let
        val z = y1 + 1
        val y2 = 1
      in
        z+y2+x
      end
  in
    f y1
  end
end
```

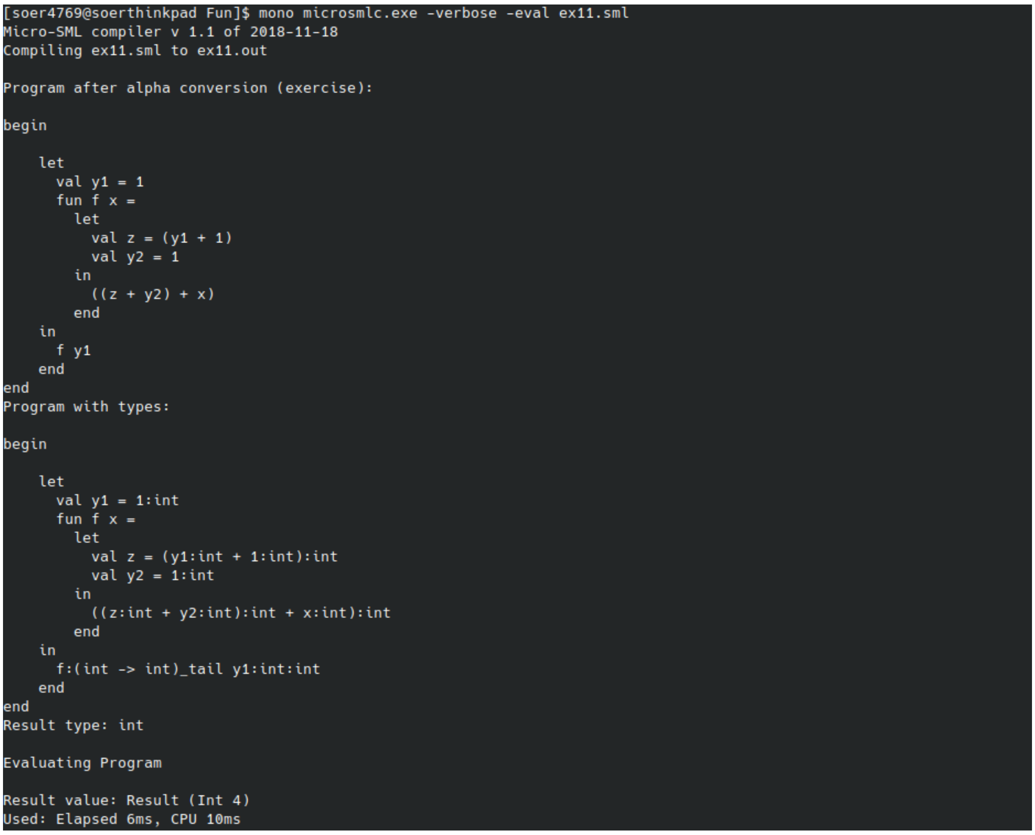
Listing 3: ex11.sml

```
begin
  let val x1 = 2
  in
    let val x2 = 5
    in
      x1 + x2
    end
  end
end
```

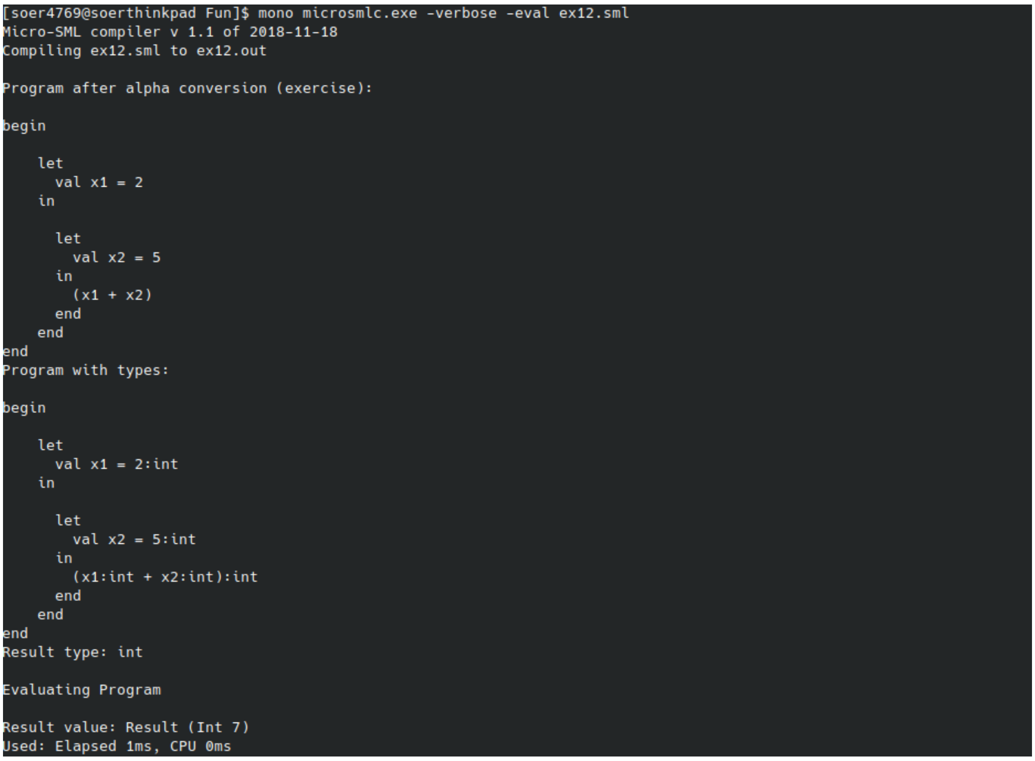
Listing 4: ex12.sml

*Refer to the files in the folder **Exercise-13.3**.*

Result:



```
[soer4769@soerthinkpad Fun]$ mono microsmlc.exe -verbose -eval ex11.sml
Micro-SML compiler v 1.1 of 2018-11-18
Compiling ex11.sml to ex11.out

Program after alpha conversion (exercise):

begin

    let
      val y1 = 1
      fun f x =
        let
          val z = (y1 + 1)
          val y2 = 1
        in
          ((z + y2) + x)
        end
    in
      f y1
    end
end
Program with types:

begin

    let
      val y1 = 1:int
      fun f x =
        let
          val z = (y1:int + 1:int):int
          val y2 = 1:int
        in
          ((z:int + y2:int):int + x:int):int
        end
    in
      f:(int -> int)_tail y1:int:int
    end
end
Result type: int

Evaluating Program

Result value: Result (Int 4)
Used: Elapsed 6ms, CPU 10ms
```



```
[soer4769@soerthinkpad Fun]$ mono microsmlc.exe -verbose -eval ex12.sml
Micro-SML compiler v 1.1 of 2018-11-18
Compiling ex12.sml to ex12.out

Program after alpha conversion (exercise):

begin

    let
      val x1 = 2
    in

      let
        val x2 = 5
      in
        (x1 + x2)
      end
    end
end
Program with types:

begin

    let
      val x1 = 2:int
    in

      let
        val x2 = 5:int
      in
        (x1:int + x2:int):int
      end
    end
end
Result type: int

Evaluating Program

Result value: Result (Int 7)
Used: Elapsed 1ms, CPU 0ms
```