T-501-FMAL Programming language
Spring 2021
Final exam, 12 April 2021
Part 2: Freetext questions

Write your answers in Icelandic or English.

In Problems 21–23, it is more important to demonstrate that you understand how to solve the problem than to resolve any errors that F# may give because you have got something wrong in the syntax.

21. (7 p) Write an F# function `group23 : 'a list -> ('a list) list` that groups the elements of a list into groups of 2 and 3 elements in alternation. Those last elements that don't make a full group should be dropped.

```
> group23 [10; 11; 12; 13; 14; 15; 16; 17; 18; 19];;
val it : int list list = [[10; 11]; [12; 13; 14]; [15; 16]; [17; 18; 19]]

> group23 [10; 11; 12; 13];;
val it : int list list = [[10; 11]]

> group23 [10; 11; 12; 13; 14; 15];;
val it : int list list = [[10; 11]; [12; 13; 14]]
```

(Hint: You may want to use mutual recursion.)

22. (6 p) Using the library functions `List.filter` and `List.map`, write an F# function `div57 : int list -> int list` that takes a list of integers, keeps those that divide by 5 or 7 (or both), and then adds 1 to each of the kept elements.

```
> div57 [1; 3; 5; 7; 8; 10; 14; 15; 21; 26; 30];;
val it : int list = [6; 8; 11; 15; 16; 22; 31]
```

23. (5 p) Write an F# function `safeFind : ('a -> bool) -> 'a list -> 'a option` that returns the first element of given list satisfying the given predicate. It should signal failure (by returning `None`) if there is no such element.

```
> safeFind (fun x -> x < 0) [16; 20; -3; 5; 4; -8; 2];;
val it : (int list) option = Some (-3)

> safeFind (fun x -> x mod 2 = 0) [5; 7; 1; 13];;
val it : (int list) option = None
```

24. (4 p) What does the expression

    `(let x = (let y = 5 in y - x) in x + y) * x`

    evaluate to in the environment `[x, 2; y, 100]`?

    Explain in detail.

    (Here and in the subsequent problems, code is given in concrete syntax. The same applies for environments. `[x, 2; y, 100]` is the environment where `x` has integer value 2 and `y` has integer value 100.)

25. (4 p) What state of the stack does the piece of stack machine code

    `[RSwap; RPop; RAdd; RMul]`

    finish in when executed from the initial stack state `[7; 6; 5; 4]`? (The top of the stack is the first element of the list.)

    Write out all intermediate states of the stack.

26. (4 p) Find a type environment (types for the free variables—all three of them) in which this expression can be typed.

```
if x < 3 && b then f x else b
```

What is the type of this expression in this type environment?

Explain.

27. (4 p) What does the expression

```
let f y = x * y in
  let x = 8 in
    let g = f in
      g x
```

evaluate to under the static scope rule in the environment `[x, 3]`?

Explain in detail.

28. (5 p) What is the most general type of the expression

```
fun x -> y, let f x = x :: x :: [] in f (f x)
```

in the type environment `[y, bool]`?

Explain in detail.

29. (4 p) Unify this pair of types, i.e., write down both their most general common substitution instance as well as the necessary type variable substitutions):

`'a -> 'b list * 'c` and `('c -> 'b) -> 'd * 'd`

(Do not rename the type variables of the two types apart before unifying.)

30. (3 p) Calculate the result of this lambda-term substitution:

$$(x\,(\lambda z.\,z\,x)\,(\lambda x.\,y\,x))\,[z\,y/x]$$

(Pay attention to which occurrences of x in the lambda-term to substitute into are free and which are bound. Also make sure you avoid variable capture.)

31. (5 p) Normalize this lambda-term (i.e., beta-reduce it in as many steps as you need until you reach a normal form) using the *leftmost-outermost* reduction strategy.

$$(\lambda x.\,(\lambda y.\,y\,(x\,y))\,(\lambda w.\,w))\,(z\,y)$$

(Again make sure you avoid variable capture.)

32. (4 p) What does the following program print?

```
void main () {
  int i;
  i = 7;
  f (&i);
  print i;
  g (i);
  print i;
}

void f (int *p) {
  *p = 3;
}

void g (int j) {
  j = 17;
}
```

Explain in detail.

33. (5 p) What does the following program print?

```
int x;
int y;
int* p;
int* q;
x = 8;
p = &x;
q = &y;
*q = *p + 3;
print y;
q = p;
x = *q * 2;
print *p;
*p = 0;
print x;
```

Explain!