# SC-T-501-FMAL Programming languages
# Final exam
# Spring 2019

Teacher: Tarmo Uustalu
Date and time: Tuesday, 16 April 2019, 9:00-12:00

Write answers in English or Icelandic.

Write clearly.

No phones, no computers.

No books, notes, no other additional material, except for a cheat sheet prepared by the teacher (attached here).

You can use additional blank paper for calculations.

This exam gives max 100 p, which will account for 60 pct of your final grade for this course.

Best of luck!

Your name: ________________________________________

Kennitala: ________________________________________

## Multiple-choice questions on all material

Encircle the correct answer. Only encircle one answer. If you encircle two or three, this is 0 p.

1. (2 p) Structural operational semantics describes
   a. how programs execute in terms of small steps between configurations
   b. which programs are syntactically correct

2. (2 p) The dynamic parent of a block is
   a. the block from which it was entered during the program's run
   b. its parent block in the syntactic structure of the program

3. (2 p) Call by value is
   a. a scope rule
   b. a parameter passing mode
   c. a compiler optimization

4. (2 p) In call by reference, in a function call
   a. the actual parameter must have an l-value
   b. can be any expression (of a type compatible with the function's type)

5. (2 p)
   a. Java uses explicit pointer types and explicit dereferencing.
   b. Java uses the "reference model" for variables of more complex types (such as objects).

6. (2 p) Tombstones is
   a. a parameter passing mode
   b. a memory safety mechanism
   c. a garbage collection technique

7. (2 p) The free list is
   a. a list to keep track of free heap memory
   b. an association list to keep track of which stack variables point where

8. (2 p) SML is a strongly typed language.
   a. No.
   b. Yes.

9. (2 p) SML supports universal parametric polymorphism.
   a. Yes.
   b. No.

10. (2 p) SML allows implicit conversions.
    a. Yes
    b. No
    c. Only from integers to reals

11. (2 p) In Prolog, operators like `+/2` and `*/2` are by default (ie always except for special contexts)
    a. uninterpreted
    b. arithmetical operators

12. (2 p) The cut predicate `!/0`

    a. cuts down the search space as it is traversed
    b. alters the order the search space is traversed
    c. neither

## Context-free grammars

13. (4 p) Consider the following context-free grammar. The terminals are $\{a, b, c\}$, the nonterminals are $\{S\}$, the start symbol is $S$. The rules are

$$\begin{array}{l} S \to aScSa \\ S \to b \end{array}$$

Write down 2 strings of 9 letters each that are derivable in this grammar and show the derivation trees.

14. (4 p) Consider the following context-free grammar. The terminals are $\{a, b\}$, the nonterminals are $\{S, A, B, C\}$, the start symbol is $S$. The rules are

$$\begin{array}{l} S \to AB \\ A \to ab \\ A \to C \\ B \to C \\ B \to ba \\ C \to a \end{array}$$

Draw two different derivation trees for the string $aba$.

## Imperative programming

15. Consider the following program in the toy language of the book:

```
{
  int x = 15;

  int p () {
    x = x + 2;
    return (5);
  }

  print (x - p ());
}
```

What is printed under
(a) (2 p) left-to-right evaluation order of expressions,

(b) (2 p) right-to-left evaluation order.

16. Consider the following program:

```
{
  int x = 1;

  void p () {
    print (x);
    x = 3;
  }

  {
    int x = 2;
    p ();
  }
  print (x);
}
```

What is printed under
(a) (3 p) the static scope rule,

(b) (3 p) the dynamic scope rule.

17. Consider this program.

```
{
  int x = 17;

  void p (int y) {
    print (y);
    y = 42;
    print (x);
 }

  p (x);
  print (x);
}
```

What is printed if parameter passing in this language is
(a) (3 p) call-by-value,

(b) (3 p) call-by-reference,

(c) (3 p) call-by-value-result.

18. (3 p) Consider this program in the version of the toy language with pointer types and explicit dereferencing:

```
{
  int* p = new int;
  int* q = new int;

  *p = 17;
  q = p;
  print (*q);
  *q = *p + 1;
  print (*p);
  p = q;
  print (*q);
}
```

What does the program print?

## Functional programming

19. (a) (3 p) Normalize this untyped lambda-term, step by step:

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

    Make sure that you do the substitutions correctly, avoiding variable capture.

    (b) (3 p) Also normalize this one.

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

20. Consider this SML program.

```
fun f g x y = if x then [g x] else y
```

(a) (3 p) What is the inferred type of `f`, i.e., its most general type?

(b) (2 p) What is the value of this expression (assuming the definition of `f` above)?

```
- f not true [false, true, false];
```

21. The library functions `map` and `filter` are defined like this.

```
fun map f [] = []
  | map f (x :: xs) = f x :: map f xs

fun filter p [] = []
  | filter p (x :: xs) = if p x then x :: filter p xs else filter p xs
```

(a) (2 p) What is the value of this expression?

```
- map (map (fn x => x + 1)) [[7,8,2],[5],[3,10,18,15]];
```

(b) (2 p) And this expression?

```
- map (filter (fn x => x < 8)) [[7,9,2],[5],[3,10,5,15]];
```

22. (4 p) Write a function `skips : 'a list -> 'a list list` that, given a list, returns the list of all lists obtainable by removing one element from the given list.

E.g.,

```
- skips ["ah", "i", "see", "this", "now"];
val it = [["i", "see", "this", "now"],
  ["ah", "see", "this", "now"],
  ["ah", "i", "this", "now"],
  ["ah", "i", "see", "now"],
  ["ah", "i", "see", "this"]] : string list list
- skips [14];
val it = [[]] : int list list
```

You can use the library function `map` (see its definition in the previous problem).

Hint: The "skips" of a cons-list are given by its tail together with the head of the list consed to each "skip" of the tail.

23. (4 p) Write a function `diffs : int list -> int list` which, given a list of integers, returns the list of differences of pairs of consecutive elements.

    On the empty list and singleton lists, it should return the empty list.

    E.g.,

```
- diffs [4,3,0,8,7,0,5];
val it = [1, 3, ~8, 1, 7, ~5] : int list

- diffs [18];
val it = [] : int list
```

## Logic programming

24. Consider the following program.

```
a(1).
a(2).
```

    What does Prolog answer to the following queries? List all answers from all successes in the order Prolog finds them (and say "false" if Prolog fails).

    Remember that `=/2` tries to unify the two arguments. `(\=)/2` is the "negation-by-failure" of `=/2`.

    (a) (2 p)

```
?- a(2).
```

    (b) (2 p)

```
?- a(1+1).
```

(c) (2 p)

```
?- a(3).
```

(d) (2 p)

```
?- a(X), X = Y.
```

(e) (2 p)

```
?- a(X), X \= 1.
```

(f) (2 p)

```
?- a(X), a(Y), X \= Y.
```

25. (4 p) Consider the following program:

```
q(a).

r(b).
r(c).

p(X) :- q(X).
p(X) :- r(X), !.
p(d).
```

What does Prolog answer to each of the following queries? List all answers in the order Prolog finds them!

(a)

```
?- p(X).
```

(b)

```
?- p(c).
```

26. (3 p) Write a predicate `ok/1` in Prolog for checking that the given term is of the form $x$`+`$y$`*`$z$ where each of $x$, $y$, $z$ is either `a` or `b`.

```
?- ok(a + b * a).
true.

?- ok(a + b * b).
true.

?- ok(a + c * b).
false.

?- ok(a * b).
false.
```

27. (4 p) Write a predicate `replaceAB/2` that replaces all occurrences of `a` as an element in a list with `b`.

```
?- replaceAB([a,b*d,c,a,e+f], Xs).
Xs = [b,b*d,c,b,e+f])

?- replaceAB([17,18,20], Xs).
Xs = [17,18,20]

?- replaceAB([4+a,a,b,c], Xs).
Xs = [4+a,b,b,c]

?- replaceAB([], Xs).
Xs = []
```