Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2018-06-02
Sal (1)	G34(7)
Tid	8-12
Kurskod	TDDA69
Provkod	TENA
Kursnamn/benämning **Provnamn/benämning**	Data- och programstrukturer Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	5
Jour/Kursansvarig Ange vem som besöker salen	Cyrille Berger
Telefon under skrivtiden	0767772870
Besöker salen ca klockan	-
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

TDDA69
Spring 2018
2018-06-02 8-12
Time Limit: 4 hours

Examiner: Cyrille Berger

This exam contains 3 pages (including this cover page) and 5 questions.
Total of points is 25p, the minimum for passing the exam is 12p, to get a four it is 16p and to get a five it is 20p.

No assistance.
Good luck!

1. (7 points) Programming paradigms and concepts.

   (a) (4 points) Draw a diagram showing the relation between the following programming paradigms:

   - First-order functional programming
   - Functional programming
   - Logic programming
   - Imperative programming
   - Sequential object-oriented programming
   - Declarative concurrent programming

   The relation between those programming paradigms could be (not all of them are necesserary usefull, and some might appear several times in the diagram):

   - +procedure
   - +closure
   - +cell(state)
   - +unification
   - +thread
   - +search
   - +port

   The diagram should be a graph where the nodes are the programming paradigms and the edges are the relations.

   (b) (3 points) Explain what tail-call optimisation is. For the following functions, tell which can be benefit from it and explain why:

```
def a(x, y):
  return b(x+y, 1)

def b(x, y):
  return a(-1, x) + y

def c(x, y):
  return x * y
```

2. (3 points) Macros.

(a) (1 point) What are macros?

(b) (1 point) What are the benefits and inconvenients of macros?

(c) (1 point) What are hygienic macros?

3. (8 points) Environment diagram.

Assume the expression below is evaluated in the order it is given.

```
function f(x)
{
  return h(g)(x+1)(4, 5);
}
function g(x)
{
  return function(y,z) { return z + (y * x); }
}
function h(f)
{
  return function(x) { return f(x+3); }
}
f(5)
```

(a) (1 point) What will the result be?

(b) (3 points) Draw a diagram that captures what is going on according to the environment model of evaluation.

(c) (2 points) Mark the important structures and explain why, and in what order, they are created and (can be) removed.

(d) (2 points) Use the diagram to show the result of the evaluation.

4. (4 points) Stack machines.

In this question, we use a stack machine with the following instruction set:

- *PUSH [constant_value]*: push the constant on the stack
- *POP [number]*: pop a certain numbers of variables from the stack
- *MUL*: pop two arguments from the stack, push the result of multiplying them
- *SUB*: pop two arguments from the stack, push the result of subtracting them
- *EQUAL*: pop two arguments from the stack, push true if they are equal, or false otherwise
- *LOAD [varname]*: push the value of variable
- *DCL [varname]*: declare the variable
- *STORE [varname]*: get the value, store the result and push the value
- *JMP [idx]*: jump to execute instruction at the given index
- *IFJMP [idx]*: pop the value and if true jump to [idx]
- *CALL [arguments]*: pop the function object and call it with the given number of arguments

- *RET*: return from a function call

(a) (2 points) Given the following factorial function:

```
var factorial = function(n)
{
  if(n == 0) {
    return 1
  } else {
    return n * factorial(n - 1);
  }
}
```

Write the list of instructions that would define the factorial function on a stack machine with the provided instruction set.

For clarity, you should provide a number for each instruction in your answer, as shown in the following example:

1. LOAD ’k’
2. PUSH ’5’
3. MUL
4. JMP 1

(b) (1 point) Write the list of instructions that would call the factorial function.

(c) (1 point) Explain what happen during a *CALL* instruction and how the *RET* instruction knows where to return.

5. (3 points) Regular expressions.

(a) (1 point) Given the following regular expression:

```
/(ab+c)*/
```

Where *+* is one or more occurence, * is zero or more occurence and *()* is used for grouping.

Which of the following strings matches:

```
var a = "abc";
var b = "ac";
var c = "";
var d = "abbbbc";
var e = "abbbcabc"
```

(b) (2 points) Explain how the regular expression is executed, using a diagram.