Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2016-10-20
Sal (1)	TER3
Tid	14-18
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	013 284023 or 076-777 28 70
Besöker salen ca klockan	ca 15:30-16
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 2016
2016-08-18 14-18
Time Limit: 4 hours

Examiner: Cyrille Berger
Tel: 076-777 28 70

This exam contains 3 pages (including this cover page) and 5 questions.
Total of points is 26p, the minimum for passing the exam is 13p, to get a four it is 17p and to get a five it is 21p.

No assistance.
Good luck!

1. (7 points) Programming paradigms.
   (a) (3 points) Explain the difference between functional programming and imperative programming.
   (b) (1 point) What is a pure function? Tell if pure and non-pure functions can be used in functional or imperative programming.
   (c) (3 points) For the following applications, select which programming paradigm (between functional or imperative) you would choose to use, and give an explanation of your choices:
      1. querying a database
      2. distributed numerical computations
      3. game

2. (2 points) Rewrite the following code using a recursion:

```
1  def compute_value(n):
2    v  = 0
3    v2 = 1
4    r  = 0
5    for l in range(0, n-1):
6      r  = v + v2
7      v  = v2
8      v2 = r
9    return r;
```

3. (8 points) Environment diagram.
   Assume the expression below is evaluated in the order it is given.

```
1  function f(x)
2  {
3    return h(g)(x+1)(4, 5);
4  }
5  function g(x)
6  {
7    return function(y,z) { return z + (y * x); }
8  }
9  function h(f)
10 {
11   return function(x) { return f(x+3); }
12 }
13 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) Logic Programming.

(a) (1 point) Give the answer(s) to the following query:

```
1  (fact (parent abraham barack))
2  (fact (parent abraham clinton))
3  (fact (parent delano herbert))
4  (fact (parent fillmore delano))
5  (fact (parent fillmore abraham))
6  (fact (parent fillmore grover))
7  (fact (grandparent (parent ?x ?y) (parent ?y ?z)))
8
9  (query (grandparent fillmore ?grandchild))
```

(b) (1 point) Explain how the query is executed.

(c) (2 points) What are the benefits and drawbacks of using logic programming languages (such as Prolog or QLog) over pure declarative programming language (such as SQL) to implement a database query language?

5. (5 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:

```
1 var factorial = function(n)
2 {
3   if(n == 0) {
4     return 1
5   } else {
6     return n * factorial(n - 1);
7   }
8 }
```

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

Write the list of instructions that would call the factorial function.

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

(c) (2 points) What is the maximum depth of the stack for a call to `function(5)` ? List all the values in the stack.