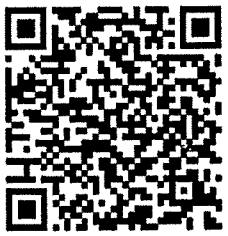
Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2017-08-17
Sal (2)	G32(3) G33(1)
Tid	14-18
Kurskod	TDDA69
Provkod	TENA
Kursnamn/benämning **Provnamn/benämning**	Data- och programstrukturer Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	7
Jour/Kursansvarig Ange vem som besöker salen	Anders Märak Leffler
Telefon under skrivtiden	073-1011291
Besöker salen ca klockan	ca kl. 16:30
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, ankn. 2362, anna.grabska.eklund@liu.se
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

Försättsblad till skriftlig tentamen vid Linköpings universitet



Datum för tentamen	2017-08-17
Sal (2)	G32(3) G33(1)
Tid	14-18
Kurskod	TDDA69
Provkod	TENA
Kursnamn/benämning **Provnamn/benämning**	Data- och programstrukturer Tentamen
Institution	IDA
Antal uppgifter som ingår i tentamen	7
Jour/Kursansvarig Ange vem som besöker salen	Anders Märak Leffler
Telefon under skrivtiden	073-1011291
Besöker salen ca klockan	ca kl. 16:30
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 2017
2017-08-17 14-18 Examiner: Cyrille Berger
Time Limit: 4 hours

This exam contains 4 pages (including this cover page) and 7 questions.
Total of points is 29p, the minimum for passing the exam is 14p, to get a four it is 19p and to get a five it is 24p.

No assistance.
Good luck!

---

1. (4 points) Programming paradigms.
   (a) (3 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

      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) (1 point) Explain the main differences between weak typing and strong typing.

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) Macros.

   (a) (2 points) What are macros?

   (b) (2 points) What are the benefits and inconvenients of macros?

5. (3 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.

6. (5 points) Concurrent Programming.

(a) (1 point) The following class defines an account:

```
1 class Account:
2   def __init__(self, balance):
3     self.balance = balance
4   def withdraw(self, amount):
5     """Withdraw money from the account."""
6       if amount > self.balance:
7           return 'Insufficient funds'
8       self.balance = self.balance - amount
```

We want to use in a multi-threaded banking system:

```
1 account = Account(100)
2 thread.start_new_thread(Account.withdraw, (account, 20))
3 thread.start_new_thread(Account.withdraw, (account, 25))
4 print(account.balance)
```

What is the expected result? Explain why with the current implementation the result can be different.

(b) (2 points) In Python, you can create a mutex with `mutex = threading.Lock()`, acquire the mutex with `mutex.acquire()` and release it with `mutex.release()`. Provide a modification of the `Account.withdraw` function to guarantee that we obtain the correct result

(c) (2 points) A common mistake with mutex is to forget to unlock it. What solution(s) would you implement in a programming language to help developers avoid this problem?

7. (3 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) (2 points) Explain how the query is executed.