5

Försättsblad till skriftlig tentamen vid Linköpings universitet

Datum för tentamen	2015-06-05
Sal (1)	KÅRA
Tid	8-12
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	Cyrille Berger
Telefon under skrivtiden	07067-772870
Besöker salen ca klockan	ca kl. 9
Kursadministratör/kontaktperson (namn + tfnr + mailaddress)	Anna Grabska Eklund, anna.grabska.eklund@liu.se, ankn. 2362
Tillåtna hjälpmedel	inga
Övrigt	
Antal exemplar i påsen	

TDDA69
Spring 2015
2015-06-05 8-12
Time Limit: 4 hours

Examiner: Cyrille Berger
Tel: 076-777 28 70

This exam contains 4 pages (including this cover page) and 7 questions.
Total of points is 39p, the minimum for passing the exam is 19p.
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. (6 points) Data structures implementation.
   (a) (2 points) Implement a dictionary structure using **only** the python **array** structure, you are **not** required to provide an optimal or efficient solution.
   You need to provide the following functions:
      1. *dict_create()*: create a new dictionary.
      2. *dict_set(d, key, value)*: associate the *value* to the *key* in the dictionary *d*.
      3. *dict_get(d, key)*: return the value corresponding to the *key* from the dictionary *d*.

Example of use:

```
h = dict_create()
dict_set(h, 'k1', 1)
dict_set(h, 'k2', 3)
print(dict_get(h, 'k2')) # Should print 3
dict_set(h, 'k2', -1)
print(dict_get(h, 'k2')) # Should print -1
```

(b) (3 points) Using the dictionary from question a, you can now implement an object system.

You need to provide the following functions:

1. *class_create(initfunc, members)*: create a new class using the *initfunc* and with the initial set of *members*. This function should return a function that can be use to instantiate new objects of the given class.
2. *class_call(obj, func, *args)*: call the function *func* of the object *obj* with the set of arguments *args*.

Example of use:

```
def car_init(obj, brand, color):
  dict_set(obj, 'color', color)
  dict_set(obj, 'brand', brand)

def car_set_speed(obj, linear_spead, angular_speed):
  dict_set(obj, 'linear_spead', linear_spead)
  dict_set(obj, 'angular_speed', angular_speed)

def car_repaint(obj, color):
  dict_set(obj, 'color', color)

car = class_create(car_init,
  [('set_speed', car_set_speed),
   ('repaint', car_repaint),
   ('linear_spead', 0.0),
   ('angular_speed', 0.0)])
my_red_volvo = car('volvo', 'red')
class_call(my_red_volvo, 'set_speed', 1.0, 3.0)
print(dict_get(my_red_volvo, 'color'))          # Should return 'red'
print(dict_get(my_red_volvo, 'linear_spead'))   # Should return 1.0
print(dict_get(my_red_volvo, 'angular_speed'))  # Should return 3.0
```

(c) (1 point) Explain in a few words what is polymorphism and how it could be implemented with the object system defined in the previous question.

4. (10 points) Environment diagram. Assume the expression below is evaluated in the order it is given.

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

(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.

(e) (2 points) Does *f(3)* always return the same value? Explain your answer.

5. (3 points) Macros.

What is printed when executing the following code?

```
1  def skipper(f, n=None):
2    if n is None:
3        return lambda n : skipper(f, n)
4    else:
5      if n % 2 == 0:
6        retval = f(n)
7      else:
8        retval = n * skipper(f, n-1)
9
10     return retval
11
12 calls = 0
13
14 @skipper
15 def fact(n):
16   global calls
17   calls += 1
18   if n < 1:
```

```
19      return 1
20    else:
21      return n * fact(n-1)
22
23  print(fact(4))
24  print(calls)
```

6. (7 points) Virtual machines.
   (a) (2 points) What are the benefits of virtual machine over tree evaluation ? And what are the drawbacks?
   (b) (3 points) How can you implement tail-call optimisation in a virtual machine?
   (c) (2 points) Explain the purpose of a garbage collector.

7. (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?