Solution Exam Programming Languages

Date: Friday, 04.06.2021.
Duration: 60 minutes
Material: You are NOT allowed to use any material (e.g., script, exercises including solutions, notes, electronic devices...)
Number of exercises: 6
Total points: 70

Firstname, lastname: _____

Matriculation number: _____

Write your name on each extra page you deliver.
Consecutively number all pages. Total number of extra pages: _____

Exercise 1 (18 Points)

Answer the following questions. Do not write more than 3 sentences each. Each question is worth 2 points.

1. Distinguish the key characteristics of the functional and logic programming styles.

2. How is `(1 + 3) * 2` computed in a stack-based language like PostScript? Note the contents of the stack after each operation.

3. Why don't pure functional languages provide loop constructs?

4. What is the difference between monomorphic and polymorphic types? Is the following Haskell function monomorphic or polymorphic? Why?

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

5. Why is normal order evaluation called lazy evaluation?
   Consider the expression: `sqr n = n * n` and compute the normal order evaluation of `sqr (7-3)`

6. What is the difference between syntax and semantics?

7. List differences between a class-based and a prototype-based programming language.

8. What happens in JavaScript if we assign a variable without using the command "var"? What problem can it cause?

9. What does the closed world assumption in Prolog state?

Answer:

1. **functional** *working in the terms of mathematical function, the state of a variable cannot be modified*

   logic *a program is decomposed into facts and rules, used to develop new facts, that is express thing that you know and rules for inferring new thing*

2. ```
   1 % 1
   3 % 1 3
   add % 4
   2 % 4 2
   mul % 8
   ```

3. *Explicit loop constructs are intended to change the state of an entity (object, variable, etc.). But in a pure functional language there is no state because of referential transparency. So iterative processes are accomplished by means of recursion, thus function calls, which does not violate referential transparency.*
   ```

4. *"Monomorphic" means "every value has a single, unique type". "Polymorphic" means that a value can have more than one type. The function is polymorphic, since it accepts a list whose elements can be any type on which the + function can be applied.*

5. *Because expressions are only evaluated when they are needed, the evaluation is delayed.*

```
sqr (7-3) => (7-3) * (7-3) => 4 * (7-3) => 4 * 4 => 16
```

6. *Syntax: the arrangement of words and phrases to create well-formed sentences in a language*
   Semantics: the meaning of a word, phrase, sentence or text
   You can create well-formed sentences (according to the syntax) that don't have a meaning (according to semantics).

7. *Class-based: works with classes, classes inherit from classes, objects cannot be extended easily, normally based on statically typed languages Prototype-based: works with prototype objects, objects inherit from objects, objects can be extended easily, normally based on interpreted and dynamically typed languages,*

8. *We override the variable in the closest scope in a scope chain. If no such variable exists, the global variable is created.*

9. *Anything that cannot be inferred with given data is assumed false.*

Exercise 2 (10 Points)

The following questions refer to the stack-based programming language PostScript.

1. (1 point) When should you use `translate` instead of `moveto`?

2. (2 points) What does the following PostScript script write on the stack?

```
/COOLFUNCTION { add 3 div 2 mul } def
8 7 COOLFUNCTION
```

3. (7 points) The following code is incomplete to output three blue boxes next to each other on the screen (See **??**). You need to complete the given code in order to fix the issue.

```
/box {
0 100 rlineto
100 0 rlineto
0 -100 rlineto
0 1 0 setrgbcolor
} def
% box accepts two parameters for x and y coordinate
100 100 box
250 100 box
showpage
```

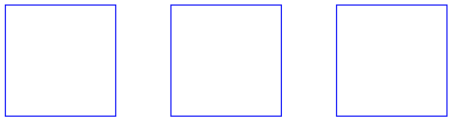

Figure 1: The expected output

Answer:

- ```
  /box {
  newpath
  moveto
  0 100 rlineto
  100 0 rlineto
  0 -100 rlineto
  closepath
  0 1 0 setrgbcolor
  fill
  } def

  100 100 box
  250 100 box
  400 100 box

  showpage
  ```
- *Translate should be used in conjunction with relative coordinates. For example, if you define a function which should draw an object starting at the center of the page (and not at the origin), you should use translate in order to move the origin to the center. Then the object is drawn relative to the new coordinate system.*
- *(8 + 7) / 3 * 2 = 15 / 3 * 2 = 5 * 2 = 10*
  ```

Exercise 3 (9 Points)

1. (3 points) Consider the following Haskell program: `[ (x+3)*x | x <- [2..6]]`
   What is the name of the syntactic sugar used? What is the output produced?

2. (6 points) Consider the following function definitions:

```
myCoolFunc a l = myCoolSubFunc 0 a l
myCoolSubFunc i n [] = []
myCoolSubFunc i n (x:xs)
    | x == n = i :  myCoolSubFunc (i+1) n xs
    | otherwise = myCoolSubFunc (i+1) n xs
```

   What is the output of the following invocations:

   (a) `myCoolFunc 5 [1..9]`

   (b) `myCoolFunc 1 []`

   (c) `myCoolFunc 2 [1, 2, 3, 3, 2, 1]`

Answer:

- *List comprehension*
  `[10,18,28,40,54]`
- 1. *`[4]`*
  2. *`[]`*
  3. *`[1, 4]`*

Exercise 4 (10 Points)

1. (4 point) Consider the following λ-expressions. Indicate which occurrences of variables are bound and which ones are free in the expressions.

   - `(λ a b . c d a b) a b (λ c d . d c)(λ e f . f) e`

   - `λ y . (λ x . z(x(λ x . y(z)))) (λ z . y(x(z)))`

2. (6 points) Reduce the following λ-expressions to their normal form whenever possible.

   - `λx. (λy . (λx. x y) x) p`

   - `(λ x . x x) (λ x . x x) y`

Answer:

- *b = bound, f = free*

```
   (\ a b . c d a b) a b (\ c d . d c) (\ e f . f) e
        | | | |  | |          | |           |  |
        f f b b  f f          b b           b  f

\ y . (\ x . z(x(\ x . y(z)))) (\ z . y(x(z)))
            | |       | |           | | |
            f b       b f           b f b
```

- $\lambda x.xp$
  p

```
——— ———
(λ x . xx) (λ x . xx) y
(λ x . xx) (λ x . xx) y
(λ x . xx) (λ x . xx) y
```

  non-terminating

Exercise 5 (10 Points)

Suppose you have a small JavaScript program:

```
var company = (function() {
    var idea = "The secret idea is X";
    return {
        budget: function () {
               return "One million dollars";
        }
    }
})()

var startup = Object.create(company);
startup.budget =  function () {
       return "Several thousand dollars";
}

var obj1 = Object.create(company);
obj1.name = "company one";
var obj2 = Object.create(company);
obj2.name = "company two";
var obj3  = Object.create(startup);
obj3.name = "startup one";
```

1. (3 points) What is the prototype of obj1, obj2 and obj3?

2. (3 points) Extend the code so that `obj1`, `obj2` and `obj3` respond to the message `bankrupt`. For example, `obj3.bankrupt()` outputs the object's name + the `"never goes bankrupt"` string, e.g., `"startup one never goes bankrupt"`.

3. (1 point) What is the output of obj3.budget()?

4. (3 points) Write a method within the closure of the `company` prototype such that the local variable `idea` will be accessible to obj1, obj2 and obj3.

Answer:

1. *company - company - startup*
2. `company.bankrupt = function () { return this.name + " never go bankrupt"; }`
3. `Several thousand dollars`
4.
```
viewidea: function()
{
return idea;
}
```

Exercise 6 (13 Points)

Create a finite collection of definite clause grammar rules to check whether a sentence is grammatically correct. A sentence can be composed of the following words:

`Subject pronoun:` `he`, `she`, `you`.

`Wh pronoun:` `what`, `where`.

`Verb:` `play`, `paint`.

`Auxiliary:` `did`.

`Verb, past tense:` `played`, `painted`.

A sentence must be in the form `subject-predicate` or `wh pronoun-auxiliary-subject pronoun-verb?`.
Predicate means either an auxiliary or verb past tense.
Note: The resulting sentence must be a simple past tense, either in the form of a sentence or question.

You can test your program with the following examples:

```
she painted // True
you painted // True
he painted // True
she played // True
she did // True
what did she paint // True
where did he play // True
he did painted // False
he what painted // False
what did she painted // False
where did you played // False
```

1. (10 points) Write all necessary definite clause grammar rules.

2. (3 points) Write a Prolog question to produce all the correct sentences in the grammar.

Answer:

```
sentence --> questionsec.

questionsec --> personal_pronoun, predicate.
questionsec --> wh, vbd, personal_pronoun, verb.

predicate --> vbd.
predicate --> verb_past.

personal_pronoun --> [she].
```

```
personal_pronoun --> [he].
personal_pronoun --> [you].
wh --> [what].
wh --> [where].
vbd --> [did].
verb --> [paint].
verb --> [play].
verb_past --> [painted].
verb_past --> [played].
```

Points

Exercise 1

Task	Points	Score
1	2	
2	2	
3	2	
4	2	
5	2	
6	2	
7	2	
8	2	
9	2	
Total	**18**	

Exercise 2

Task	Points	Score
1	1	
2	2	
3	7	
Total	**10**	

Exercise 3

Task	Points	Score
1	3	
2	6	
Total	**9**	

Exercise 4

Task	Points	Score
1	4	
2	6	
Total	**10**	

Exercise 5

Task	Points	Score
1	3	
2	3	
3	1	
4	3	
Total	**10**	

Exercise 6

Task	Points	Score
1	10	
1	3	
Total	**13**	

TOTAL

Exercise	Points	Score
1	18	
2	10	
3	9	
4	10	
5	10	
6	13	
Total	**70**	