Pearls of Computer Science - Pearl 011 - Practice exam

Course: B-CS-MOD01-1A-202001022 B-CS Pearls of Computer Science Core 202001022

Contents: Pages:

Number of questions:	30
Generated on:	Sep 17, 2020

Pearls of Computer Science - Pearl 011 - Practice exam

Course: B-CS Pearls of Computer Science Core 202001022

- Time: 14:45 - 15:45 (60 minutes)
    - You can start the exam when it is made available. You cannot start earlier.
    - If you finish before 15:30, you can quietly leave (**do not leave between 15:30 and 15:45)**
    - **After 15:45 remain seated; we notify when you can leave**
- As a reference you may use “Haskell operators and functions” (click here: See attachment:) *and* your notes (one A4, double sided) during the exam.
- Next to what was provided by us, you are only allowed to have this on your table:
    - Your **student card**; please put it clearly visible on the right side of the table (such that an invigilator does not have to touch it);
        - If you do not have a student card yet: use a certified photo ID (ID card / passport opened on the photo page/ drivers licence)
        - If you have a card granting you extra time, place it next to it.
    - One A4 with notes (double sided).
    - A pen or a pencil(+eraser)
- All other things (e.g. calculators, laptops, mobile phones, smart watches, books etc.) are not allowed.
  Put those in your bag now (switched off)!
- A simple calculator is provided within the test environment (but likely not needed).
- Scrap paper is provided. In addition, you may use "txt", a small notepad application on the Chromebooks (small icon at the bottom of the screen).
- The answers to the questions (a/b/c/d) may be shuffled for some questions. The order may be different for individual students.
- Code is sometimes formatted **`bold`** within text to make it better readable.
- 30 multiple choice questions (weighted equally), correction for guessing is used.
- After a post-exam analysis, questions that turned out to be badly formulated or too difficult may be removed by the corrector. This may influence your grade.

Number of questions: 30

You can score a total of 30 points for this exam, you need 18.75 points to pass the exam.

1 Which of the Haskell functions below is a correct implementation of the mathematical function
1 pt. $f(x) = x^2 + x + 3$?

```
f1 :: Double -> Double
f1 (double x) = x^2 + x + 3

f2 :: Double -> Double -> Double
f2 x = x^2 + x + 3
```

a. Only **`f1`** is a correct implementation.

b. Only **`f2`** is a correct implementation.

c. **`f1`** and **`f2`** are both correct implementations.

d. **`f1`** and **`f2`** are both incorrect implementations.

2 Which of the Haskell functions below can calculate the mathematical function
1 pt. $f(a, b) = \sqrt{a^2 + b^2}$?

```
nrm1 :: Double -> Double -> Double
nrm1 a b = sqrt (a^2 + b^2)

nrm2 :: (Double, Double) -> Double
nrm2 (a, b) = sqrt (a^2 + b^2)
```

a. Only **`nrm1`** is a correct implementation.

b. Only **`nrm2`** is a correct implementation.

c. **`nrm1`** and **`nrm2`** are both correct implementations.

d. **`nrm1`** and **`nrm2`** are both incorrect implementations.

3 Is the

1 pt.

```
else...
```

part in a Haskell **`if`** expression mandatory?

a. No, since the **`else...`** part may sometimes be irrelevant and left out.

b. Haskell does not support **`if`** and/or **`else`** at all; conditions are checked using pattern matching and/or guards.

c. No, since the **`else...`** part can be taken care of with pattern matching and/or guards.

d. Yes, since Haskell is a functional language the function must always produce a result.

4 When applying guards, **`otherwise`** is convenient to use. For example:

1 pt.

```
max x y | x < y     = y
        | otherwise = x
```

`otherwise` is not a reserved keyword in Haskell, but defined as a function. How is **`otherwise`** defined?

a. otherwise = (==)

b. otherwise = False

c. otherwise = True

d. otherwise x = x

5 What is the type of the following function?

1 pt.

```
h ("*",y)= (y, (y,y))
h (x,y)  = (y, (x,y))
```

a.
```
h :: (a, a) -> (a, a)
```

b.
```
h :: [a] -> [[a]]
```

c.
```
h :: ([Char], a) -> (a, ([Char], [Char]))
```

d.
```
h :: ([Char], [Char]) -> ([Char], ([Char], [Char]))
```

6 A data type is needed to describe the combination of a name (e.g., `"John"`) and an age (e.g., `42`). Which of the following two methods can be used to accomplish this?

1 pt.

1. `("John", 42)`

2. `["John", 42]`

a. Only 1 is correct.

b. Only 2 is correct.

c. Both 1 and 2 are correct.

d. 1 and 2 are both incorrect.

7 Which of the following expressions is *incorrect*?

1 pt.

a.

```
[] : [[]] : [[[]]]
```

b.

```
’d’ : "abc"
```

c.

```
1:2:3:4:5:6:[]
```

d.

```
[(1,2), (3,4)] ++ [(’1’,’2’)]
```

8 Which function retrieves the *second* element from a *list* of at least two elements?

1 pt.

a.

```
second (x:y:xs) = y
```

b.

```
second xs = take 2 xs
```

c.

```
second xs = snd xs
```

d.

```
second xs = drop 1 xs
```

9
1 pt.

Which of the definitions below is correct Haskell code and can be used to extract the first value from a 3-tuple? (e.g., **`first (1,2,3)`** results in **`1`**)

a.
```
first (a:b:c) = a
```

b.
```
first x = fst x
```

c.
```
first xs = head xs
```

d.
```
first (a,b,c) = a
```

10
1 pt.

Is the following definition correct?

```
ones = 1 : ones
```

a. No, **`ones`** is defined in terms of itself. This is not allowed.

b. No, **`ones`** is not properly defined since it receives no function arguments.

c. No, a number is put in front of a function, which is incorrect.

d. Yes.

11
1 pt.

Is the following function correct (i.e., accepted by the Haskell compiler)?

```
greaterthan n = (n+1) : greaterthan (n+1)
```

a. No, the recursive step uses increasing numbers instead of decreasing numbers. This is not allowed.

b. No, the calculation never terminates and is therefore not accepted.

c. No, this definition results in a typing error.

d. Yes.

12 Consider the following function:

1 pt.

```
fun []       = []
fun [x]      = [x]
fun (x:y:xs) = y : fun (x:xs)
```

What is the result of the following expression?

```
fun "abcde"
```

a.
`"bcdea"`

b.
`"abcde"`

c.
`"eabcd"`

d.
`"edcba"`

13 Consider the following functions:
1 pt.

```
fn' [x]     = []
fn' (x:xs) = x : fn' xs

fn [x]      = x
fn [x,y]    = x
fn (x:xs)  = fn (fn' xs)
```

What is the result of the following expression?

```
fn "word"
```

a.
`"word"`

b.
`’o’`

c.
`"drow"`

d.
`’r’`

14 Consider the following function:

1 pt.

```
fun [] = []
fun [x] = [x]
fun (x:xs) = x : fun (tail xs)
```

What is the result of the following expression?

```
fun "abcde"
```

a.
`"bd"`

b.
`"ace"`

c.
`"abcde"`

d.
`"bcde"`

15 Consider the following function:

1 pt.

```
g [] = []
g [x] = [x]
g (x:x':xs) = g xs ++ [x]
```

What is the result of the following expression?

```
g [1,2,3,4,5,6]
```

a.
`[1,3,5]`

b.
`[5,3,1]`

c.
`[6,5,4,3,2,1]`

d.
`[2,3,4,5,6,1]`

16 What result is produced by the following Haskell code?

1 pt.

```
f = [ max x y | x <- [1..3], y <- [1..3] ]
```

a.

```
[1,2,2,3,3,3]
```

b.

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

c.

```
[1,2,3,2,2,3,3,3,3]
```

d.

```
[3,3,3,3,3,3,3,3,3]
```

17 What result is produced by the following Haskell code?

1 pt.

```
[ (y,x) | x <- "abc", y <- [1,2,3] ]
```

a.

```
[(1,’a’),(1,’b’),(1,’c’),(2,’a’),(2,’b’),(2,’c’),(3,’a’),(3,’b’),(3,’c
’)]
```

b.

```
[(1,’a’),(2,’a’),(3,’a’),(1,’b’),(2,’b’),(3,’b’),(1,’c’),(2,’c’),(3,’c
’)]
```

c.

```
 [(1,’a’),(2,’b’),(3,’c’)]
```

d.

```
[(1, "abc"),(2,"abc"),(3,"abc")]
```

18 Consider the following function:

1 pt.

```
w u = [ n * m | n <- [1..u], m <- [n..u], n * m <= u ]
```

What result is produced by the following expression?

```
w 4
```

a.

```
[1,2,3,4,4,6,8,9,12,16]
```

b.

```
[1,2,3,4]
```

c.

```
[1,2,3,4,4]
```

d.

```
[1,2,3,4,2,4,3,4]
```

19 What result is produced by the following Haskell code?

1 pt.

```
f = [ (x,y) | x <- [1..4], y <- [1..x], x + y == 5]
```

a.

```
[(3,2), (4,1)]
```

b.

```
[(1,4), (2,3), (3,2), (4,1)]
```

c.

```
[(1,4), (2,3)]
```

d.

```
(1,4)
```

20 Which of the functions below is correctly defined *and* returns **`True`** when the list contains exactly two values, and **`False`** otherwise?

1 pt.

```
hasTwo1 :: [a] -> Bool
hasTwo1 (x:y:[]) = True
hasTwo1 xs = False

hasTwo2 :: [a] -> Bool
hasTwo2 xs = False
hasTwo2 [x,y] = True
hasTwo2 [x] = False
```

a. Only **`hasTwo1`** is correct.

b. Only **`hasTwo2`** is correct.

c. Both **`hasTwo1`** and **`hasTwo2`** are correct.

d. **`hasTwo1`** and **`hasTwo2`** are both incorrect.

21 Which of the two functions below result in **`True`** when its argument is a list and **`False`** for *all* other types (e.g., **`Int`**)?

1 pt.

```
isList1 [xs] = True
isList1 xs = False

isList2 [] = True
isList2 xs = False
```

a. Only **`isList1`** is correct.

b. Only **`isList2`** is correct.

c. Both **`isList1`** and **`isList2`** are correct.

d. **`isList1`** and **`isList2`** are both incorrect.

22 Which of the definitions below are correctly defined *and* result in **`False`** when both its (**`Bool`**) arguments are the same, and **`True`** otherwise?

1 pt.

```
xor2  True  True  = False
xor2  False False = False
xor2  x     y     = True

xor2' x     x     = False
xor2' x     y     = True
```

a. Only **`xor2`** is correct.

b. Only **`xor2'`** is correct.

c. Both **`xor2`** and **`xor2'`** are correct.

d. **`xor2`** and **`xor2'`** are both incorrect.

23 Which of the definitions below are correctly defined, *and* results in **`True`** when at least one of its inputs is **`True`** and results in **`False`** otherwise?

1 pt.

```
or2  True  y     = True
or2  x     True  = True
or2  x     y     = False

or2' False False = False
or2' x     y     = True
```

a. Only **`or2`** is correct.

b. Only **`or2'`** is correct.

c. Both **`or2`** and **`or2'`** are correct.

d. **`or2`** and **`or2'`** are both incorrect.

24 The function **`concat`** concatenates all the (zero or more) lists within a list to a single list, e.g.,

1 pt.

```
concat ["ab", "cde", "f"] = "abcdef"
```

Which function below correctly implements **`concat`**?

a.

```
concat []     = []
concat (x:xs) = x : concat xs
```

b.

```
concat []     = []
concat (x:xs) = x ++ concat xs
```

c.

```
concat [[]]   = []
concat (x:xs) = x : concat xs
```

d.

```
concat [[]]   = []
concat (x:xs) = x ++ concat xs
```

25 The Haskell function **`unwords`** glues together a list of words into a string (**`[Char]`**) wherein the words are separated by a space. For example:
1 pt.

```
unwords ["Haskell", "is", "fun"] = "Haskell is fun"
```

Which of the functions below provides a correct implementation of **`unwords`**?

a.

```
unwords []     = []
unwords [x]    = [x]
unwords (x:xs) = x : " " ++ (unwords xs)
```

b.

```
unwords []     = []
unwords [x]    = x
unwords (x:xs) = x : " " ++ (unwords xs)
```

c.

```
unwords []     = []
unwords [x]    = [x]
unwords (x:xs) = x ++ " " ++ (unwords xs)
```

d.

```
unwords []     = []
unwords [x]    = x
unwords (x:xs) = x ++ " " ++ (unwords xs)
```

26 Which of the functions below increases all the elements within a list by one?
1 pt.

a.

```
inc xs = xs + 1
```

b.

```
inc xs = map (++) xs
```

c.

```
inc xs = map (+1) xs
```

d.

```
inc xs = map (+) xs
```

27 Which of the following functions is correct and tests whether a nonempty list is nondecreasing, i.e., **`b >= a`** whenever **`a`** comes before **`b`** in the list?

1 pt.

a.

```
nondec (x:xs) = and (zipWith (>=) (xs) (x:xs))
```

b.

```
nondec (x:xs) = and (map (>=) (xs, (x:xs)))
```

c.

```
nondec (x:xs) = and (zipWith (xs) >= (x:xs))
```

d.

```
nondec (x:xs) = and (map (xs >= (x:xs)))
```

28 What is the type of the following function?

1 pt.

```
h []  = []:[]
h [x] = [x,x]
h xs  = xs
```

a.

```
h :: [a] -> [a]
```

b.

```
h :: [[a]] -> [a]
```

c.

```
h :: [a] -> [[a]]
```

d.

```
h :: [[a]] -> [[a]]
```

29 The predefined Haskell function **`subtract`** subtracts its first argument from the second. For example:

1 pt.

```
subtract 5 3 = -2
```

This is awkward when the function is used as an infix operator, e.g.:

```
5 ‘subtract‘ 3 = -2
```

To “fix” this, we may define some new function that we call **`subtract'`**, for which we have its arguments
flipped, i.e., **`5 ‘subtract′‘ 3 = 2`**. The higher order function **`flip`** is used define **`subtract'`** as:

```
subtract′ = flip subtract
```

Which of the functions below correctly implements the **`flip`** function to obtain the behavior described above?

a.

```
    flip x y = (y, x)
```

b.

```
    flip f x y = f (y,x)
```

c.

```
    flip f x y = f y x
```

d.

```
    flip x y = y x
```

30 The Haskell functions **`min`** and **`max`** determine respectively the minimum and maximum out of two values. We aim to write a function **`select`** that generalizes this behavior:

1 pt.

```
select :: (a->a->Bool) -> a -> a -> a
```

Using **`select`**, we may e.g. define **`min`** as

```
min a b = select (<) a b
```

Which of the functions below provides a correct implementation of **`select`**?

a.
```
    select f a b = [ x | x <- [a,b], f x ]
```

b.
```
    select f a b = if f a b then a else b
```

c.
```
    select f a b = f a b
```

d.
```
    a ‘select‘ b = f
```

Name:				Signature:
Date:	/ /	Birth date:	/ /	

Course: B-CS-MOD01-1A-202001022 B-CS Pearls of Computer Science Core 202001022 - Questions: Pearls of Computer Science - Pearl 011 - Practice exam

- Time: 14:45 - 15:45 (60 minutes)
  - You can start the exam when it is made available. You cannot start earlier.
  - If you finish before 15:30, you can quietly leave (**do not leave between 15:30 and 15:45)**
  - **After 15:45 remain seated; we notify when you can leave**
- As a reference you may use “Haskell operators and functions” (click here: See attachment:) *and* your notes (one A4, double sided) during the exam.
- Next to what was provided by us, you are only allowed to have this on your table:
  - Your **student card**; please put it clearly visible on the right side of the table (such that an invigilator does not have to touch it);
    - If you do not have a student card yet: use a certified photo ID (ID card / passport opened on the photo page/ drivers licence)
    - If you have a card granting you extra time, place it next to it.
  - One A4 with notes (double sided).
  - A pen or a pencil(+eraser)
- All other things (e.g. calculators, laptops, mobile phones, smart watches, books etc.) are not allowed.
  Put those in your bag now (switched off)!
- A simple calculator is provided within the test environment (but likely not needed).
- Scrap paper is provided. In addition, you may use "txt", a small notepad application on the Chromebooks (small icon at the bottom of the screen).
- The answers to the questions (a/b/c/d) may be shuffled for some questions. The order may be different for individual students.
- Code is sometimes formatted `bold` within text to make it better readable.
- 30 multiple choice questions (weighted equally), correction for guessing is used.
- After a post-exam analysis, questions that turned out to be badly formulated or too difficult may be removed by the corrector. This may influence your grade.

	A	B	C	D			A	B	C	D
1 1 pt.	○	○	○	○		**4** 1 pt.	○	○	○	○
2 1 pt.	○	○	○	○		**5** 1 pt.	○	○	○	○
3 1 pt.	○	○	○	○		**6** 1 pt.	○	○	○	○

7 1 pt.	A ○	B ○	C ○	D ○
8 1 pt.	A ○	B ○	C ○	D ○
9 1 pt.	A ○	B ○	C ○	D ○
10 1 pt.	A ○	B ○	C ○	D ○
11 1 pt.	A ○	B ○	C ○	D ○
12 1 pt.	A ○	B ○	C ○	D ○
13 1 pt.	A ○	B ○	C ○	D ○
14 1 pt.	A ○	B ○	C ○	D ○
15 1 pt.	A ○	B ○	C ○	D ○
16 1 pt.	A ○	B ○	C ○	D ○
17 1 pt.	A ○	B ○	C ○	D ○
18 1 pt.	A ○	B ○	C ○	D ○
19 1 pt.	A ○	B ○	C ○	D ○
20 1 pt.	A ○	B ○	C ○	D ○
21 1 pt.	A ○	B ○	C ○	D ○
22 1 pt.	A ○	B ○	C ○	D ○
23 1 pt.	A ○	B ○	C ○	D ○
24 1 pt.	A ○	B ○	C ○	D ○
25 1 pt.	A ○	B ○	C ○	D ○
26 1 pt.	A ○	B ○	C ○	D ○
27 1 pt.	A ○	B ○	C ○	D ○
28 1 pt.	A ○	B ○	C ○	D ○
29 1 pt.	A ○	B ○	C ○	D ○
30 1 pt.	A ○	B ○	C ○	D ○

Correction model

1. D
1 pt.

2. C
1 pt.

3. D
1 pt.

4. C
1 pt.

5. D
1 pt.

6. A
1 pt.

7. D
1 pt.

8. A
1 pt.

9. D
1 pt.

10. D
1 pt.

11. D
1 pt.

12. A
1 pt.

13. B
1 pt.

14. B
1 pt.

15. B
1 pt.

16. C
1 pt.

17. B
1 pt.

18. C
1 pt.

19. A
1 pt.

20. A
1 pt.

21. D
1 pt.

22. A
1 pt.

23. C
1 pt.

24. B
1 pt.

25. D
1 pt.

26. C
1 pt.

27. A
1 pt.

28. D
1 pt.

29. C
1 pt.

30. B
1 pt.

Caesura

Applied guessing score: 7.5 pt

Points scored	Grade
30	10
29	9.6
28	9.2
27	8.8
26	8.4
25	8.0
24	7.6
23	7.2
22	6.8
21	6.4
20	6.0
19	5.6
18	5.2
17	4.8
16	4.4
15	4.0
14	3.6
13	3.2
12	2.8
11	2.4
10	2.0
9	1.6
8	1.2
7	1.0
6	1.0
5	1.0
4	1.0
3	1.0
2	1.0
1	1.0
0	1.0

Question identifiers

These identifiers can be used to track the exact origin of the question. Use these identifiers together with the identifier of this document when sending in comments about the questions, so that your comment can be connected precisely with the question you are referring to.

Document identifier: 1495-3607

Question number	Question identifier	Version identifier
1	16057	066ca6c3-8170-ebe3-53f8-584af6ba205b
2	16060	d0a150dd-28b6-346a-110a-741c03b6a5dd
3	16063	a81fb356-ee1c-c7f4-662a-eba41f0a53b8
4	16066	ed26e612-100e-29fa-56da-c3db23ebeea7
5	16069	f4d02b8a-c95e-5a93-0486-b1fb50d86658
6	16072	5d3669b9-44dd-52c4-4b8d-0045b1812032
7	16075	9a8ea4fa-73b9-87a5-146e-a73dcac12213
8	16078	c126ee7f-683d-c084-1f31-7d33db2d29fa
9	16081	63f95d44-516b-e572-172d-c52da584bc66
10	16084	c86fd4e3-c6fc-b8ee-7c7e-61a124110779
11	16087	5bfeef43-bfca-3812-51a7-afc9ac78d6ed
12	16090	552027c8-d40a-4ec0-3e80-0d920f613a6d
13	16093	cc27f9f0-61c6-504f-0ece-c26a6dda42ee
14	16096	5ad3e3c5-803b-1c18-04c3-633db1f21ccc
15	16099	5d6f9241-381a-5d66-bc48-5e92b58c928a
16	16102	d2cd3771-9fc3-74ea-9a89-3dd10379bdc8
17	16105	2e62901a-de22-5f11-0b67-6bb2e262ca28
18	16108	d5c4880b-0fff-d4ea-9408-070ae5a8d45c
19	16111	f36ea41b-bcc0-a5fe-cd41-c485eb3ddcda
20	16114	8e73b8cf-a287-fae1-8004-d86218dec94c
21	16117	aac51f8b-0e94-8491-bc21-7fac738eea2d
22	16120	79707263-4020-6a03-fde7-3056aa4faf2d
23	16123	2ad4d543-5b83-bdd0-36ed-da62fa21904e
24	16126	fcdf502b-1784-f261-386e-4dba016d7d6b
25	16129	ad7b9193-1378-d2c8-db2b-4fe9537ccc4a
26	16132	e767be76-9c8b-f878-2093-011f2b5999e5
27	16135	f58b23cb-14fa-d2ab-8187-1bce428f19ac

28	16138	93ae26ec-d25b-a2a2-a175-999b47b96599
29	16141	2f68b96f-b321-3e4a-4812-9ae45ce3f1a6
30	16144	ea5003cd-59d8-b4ce-faab-d820d1380f0c