Final Exam (Part 1) in Program Design and Data Structures (1DL201)

Teachers: Tjark Weber and Johannes Borgström

2020-01-10 / 08:00 - 13:00

Instructions

Read and follow these instructions carefully to increase your chance of getting good marks.

- This is a closed book exam. You may use a standard English dictionary. Otherwise, **no notes, calculators, mobile phones, or other electronic devices are allowed.** Cheating will not be tolerated.
- This is a multiple-choice exam. There are **twenty** questions. Each question has exactly **one** correct answer.
- Note your responses on the question sheets, and transfer them to the answer sheet only when you are ready to hand in.
- Read the instructions on the answer sheet before you start.
- You may keep these question sheets. **Only hand in the answer sheet.**
- Johannes will come to the exam hall around 9:00 to answer questions.

Good luck!

Questions

Please choose a single answer for each question. Read the questions carefully, and watch out for negations (**not**, **except**, etc.).

```
{- func m n
   PRE: ?PRE?
   RETURNS: ?RETURNS?
 -}

-- VARIANT: ?VARIANT?

func :: ?TYPE?
func m 0 = m
func m n = func (m+1) (n-1)
```

Question 1: What is the value of `func 3 4` ?

- A `3`
- B `4`
- C `[3,4]`
- D `7`
- E `12`

Question 2: What is a type (`?TYPE?`) of `func`?

- A `Int -> Int -> Int`
- B `(Int,Int) -> Int`
- C `a -> a -> a`
- D `(a -> a) -> a`
- E `[Int] -> Int`

Question 3: What is the most appropriate precondition (`?PRE?`) for `func m n`?

- A `m ≥ 0`
- B `True`
- C `m ≥ n`
- D `n ≥ 0`
- E `m+n`

Question 4: What is an appropriate description of the return value (`?RETURNS?`) for `func m n`?

- A `m`
- B `n`
- C `m+n`
- D `func (m+1) (n-1)`
- E `m*n`

Question 5: Which of the following is a variant (`?VARIANT?`) for the function `func m n`?

- A `m+n`
- B `m`
- C `n`
- D `m*n`
- E `1`

Question 6: Which of the following expressions, when evaluated, will result in a run-time error?

- A `` False && (1 `div` 0 > 0) ``
- B `` True || (1 `div` 0 < 0) ``
- C `` let x = 1 `div` 0 in 42 ``
- D `` case 1 `div` 0 of _ -> 42 ``
- E None of the above.

Question 7: Consider the following declarations. Which one does **not** contain an example of shadowing?

- A `f x =  \x -> x`
- B `f x =  let x = x in 0`
- C `f x =  case x of _ -> x`
- D `f x =  let x = 0 in x`
- E `f x =  case x of x -> x`

Question 8: Recall the algorithm for computing the greatest common divisor (GCD) of two non-negative numbers `a` and `b`:

- If `b == 0` then the GCD is `a`.
- Otherwise, it is computed as the GCD of `b` and `` a `mod` b ``.

This algorithm is an example of a particular form of recursion. Which?

- A Simple recursion
- B Complete recursion
- C Multiple recursion
- D Mutual recursion
- E None of the above.

Question 9: Consider the function

```
sums :: Num a => [a] -> [a]
sums [] = [0]
sums l  = (sum l) : (sums (tail l))
```

Which is the best description of the return value (`RETURNS`) for `sums l`?

- A an increasing list of numbers
- B the sums of the suffixes of `l`
- C a list with the same type as `l`
- D a list of sums of numbers in `l`
- E None of the above.

Question 10: What is the worst-case time complexity of `sums l`, where $n =$ `length l`?

- A $\mathrm{O}(n^2)$
- B $\mathrm{O}(n)$
- C $\Theta(n \log n)$
- D $\Omega(n^2 \log n)$
- E None of the above.

Question 11: Which of the following is a type of `map foldl` ?

- A `[a -> b -> b] -> [b] -> [a] -> [b]`
- B `[a -> b -> a] -> [a] -> [b] -> [a]`
- C `[a -> b -> a] -> [a -> [b] -> a]`
- D `[a -> b -> b] -> [b -> [a] -> b]`
- E `[a -> b -> a] -> [a] -> [[b] -> a]`

Question 12: What is the result of evaluating the following comprehension?

```
[x+2*y | x <- [1..6], y <- [2,5], x >= y]
```

- A `[7,8,9,10,16]`
- B `[6,7,8,9,15,10,16]`
- C `[6,7,8,9,10,15,16]`
- D An error
- E None of the above.

Question 13: Recall the insertion sort algorithm. Suppose the algorithm is applied to the input list `[8,12,6,3,19,4,6,2]`. What is the remaining list and the intermediate sorted list after sorting four items?

- A `[8,12,6,3]` and `[2,4,6,19]`
- B `[2,4,6,19]` and `[3,6,8,12]`
- C `[19,4,6,2]` and `[3,6,8,12]`
- D `[8,12,19,16]` and `[2,3,4,6]`
- E `[19,4,6,2]` and `[8,12,6,3]`

Question 14: Fill in the missing part (`...`) in the following definition.

For non-negative functions f and g, $f(n) = \Theta(g(n))$ if and only if `...`:

$$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n).$$

- A there exist $n_0 \geq 0$ and c_1, $c_2 > 0$ such that for some $n > n_0$
- B there exists $n_0 \geq 0$ such that for all c_1, $c_2 > 0$ and all $n > n_0$
- C there exist $n_0 \geq 0$ and c_1, $c_2 > 0$ such that for all $n > n_0$
- D for all $n_0 \geq 0$ there exist c_1, $c_2 > 0$ such that for all $n > n_0$
- E for all $n_0 \geq 0$ there exist c_1, $c_2 > 0$ such that for some $n > n_0$

Question 15: The run-time cost of which of the following functions is **not** described by the recurrence $T(n) = T(n-1) + \Theta(1)$? (Assume that each function has a suitably defined base case with constant run-time cost, and that n gives the size of the function's argument.)

A `f n = f (n-1) - 1`

B `f n = (n-1) + f 1`

C `f n = f (n-1) + 1`

D `f (x:_:xs) = f (x:xs)`

E `f (_:xs) = f xs + 1`

Question 16: Consider the following recurrence.

$$\begin{aligned} C(0) &= 1 \\ C(n) &= 2 \cdot C(n-1) + \log n \end{aligned}$$

According to the "Doctor Theorem" the closed form of this recurrence is

A $C(n) = \Theta(2^n)$

B $C(n) = 1, 2, 5, 11.58, \ldots$

C $C(n) = \Theta(2^n \log n)$

D $C(n) = \Theta(n)$

E The "Doctor Theorem" does not apply.

Question 17: Suppose `f` is a function of type `(a -> a) -> a -> a`. Which of the following expressions is **not** type correct?

A `f (\x -> x)`

B `f (+3)`

C `f map`

D `f f`

E `f . f`

Question 18: Given these declarations,

```
f x y z = x z || y z
b = (<=5) . abs
```

which of the following expressions is **not** well-typed?

A `foldl f b`

B `foldr f b`

C `map f`

D `filter b`

E None of the above.

Question 19: Which of the following expressions does **not** evaluate to the same value as the other four?

- A `[0,3-2,3+2,3*2]`
- B `map (\x -> mod (x*x*x + 1) 12) [11,0,10,5]`
- C `[x*y + 3 | x <- [-1,1], y <- [2,3]]`
- D `[x + y | x <- [0,5], y <- [0,1]]`
- E `map (+3) (filter ((>1).abs) [-3..3])`

Question 20: Suppose you want to represent general trees (i.e., trees with arbitrary out-degree), where each node carries a label of type `[Int]` and a list of children. For instance,

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

would be an example of such a tree.

Which of the following type declarations does **not** generate a type that contains the above tree value?

- A `data Tree = Node [Int] [Tree]`
- B `data Tree a = Node a [Tree a]`
- C `data Tree a = Node [a] [Tree a]`
- D `data Tree a = Node a [Tree [a]]`
- E `data Tree a = Node [Int] [Tree a]`