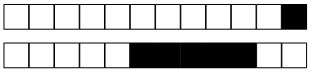

+1/1/60+

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

Teachers: E. Castegren, A. Jimborean, T. Weber

2016-12-20 / 8:00–13:00

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. Each question has exactly **one** correct answer.
- You may keep these question sheets. **Only hand in the answer sheet.** Also read the instructions on the answer sheet before you start.
- Tjark Weber will come to the exam hall around 10:00 to answer questions.

Good luck!

Master Theorem

Given a recurrence of the form

$$T(n) = aT(n/b) + f(n)$$

Case 1: If $f(n) = O(n^c)$ where $c < \log_b a$
then $T(n) = \Theta(n^{\log_b a})$.

Case 2: If $f(n) = \Theta(n^c (\log n)^k)$ where $c = \log_b a$ and $k \geq 0$
then $T(n) = \Theta(n^c (\log n)^{k+1})$.

Case 3: If $f(n) = \Omega(n^c)$ where $c > \log_b a$ and the regularity condition holds
then $T(n) = \Theta(f(n))$.
The regularity condition is that for some constant $r < 1$, $a \cdot f(n/b) \leq r \cdot f(n)$ for all sufficiently large n.

Common Material

Some of the exam questions refer to the following function:

```
{- split ls
   PRE: ?PRE?
   POST: ?POST?
 -}
split :: ?TYPE?
-- VARIANT: ?VARIANT?
split (x:y:ls) = let (xs,ys) = split ls in (x:xs, y:ys)
split ls       = (ls, [])
```

Questions

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

Question 1: What is the value of `split [1,2,3,4]` ?

A `[(1,2),(3,4)]`
B `([1,2],[3,4])`
C `([1,3],[2,4])`
D `10`
E `[1,3]`

Question 2: What is the type (`?TYPE?`) of `split`?

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

Question 3: What is the most appropriate precondition (`?PRE?`) for `split ls`?

A `True`
B `ls` is a list
C `length ls` ≥ 2
D `ls` is non-empty
E `False`

Question 4: What is the most appropriate postcondition (`?POST?`) for `split ls`?

A a pair `(xs,ys)` such that `xs++ys` is a permutation of `ls`
B a list of all pairs in `ls`
C `True`
D `(ls, [])`
E `(x:xs, y:ys)`

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

A `length ls` ≥ 2
B $\frac{1}{2} \cdot$ (`length ls`)
C `0`
D `length ls`
E `x + y + length ls`

Question 6: Which of the following statements is **false**?

A `split` is a polymorphic function.

B `split` is a higher-order function.

C `split` is a recursive function.

D `split` terminates for all valid arguments.

E The definition of `split` is type-correct.

Question 7: Consider this alternative definition of `split`, where the order of the two equations has been swapped:

```
split ls       = (ls, [])
split (x:y:ls) = let (xs,ys) = split ls in (x:xs, y:ys)
```

Which of the following statements is correct for this definition of `split`?

A If `split ls` returns the pair `(xs,ys)`, then `xs++ys` is equal to `ls`.

B Evaluating `split ls` does not terminate.

C This definition is not type-correct.

D This definition of `split` is equivalent to the definition given earlier (i.e., the two functions behave exactly the same).

E If `split ls` returns the pair `(xs,ys)`, then `xs` and `ys` have the same length.

Question 8: Consider the function

```
f x = div 0 x + div x 0
```

Evaluating `f 42` will result in ...

A `Infinity`

B a syntax error.

C a type error.

D none of these.

E a run-time error.

Question 9: Which of the following values does **not** match the pattern `(_, [2,x])` ?

A `(1, [2])`

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

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

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

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

Question 10: Consider the function

```
f x = let f x = x+1 in f (f x)
```

What is the value of `let x=0 in f x` ?

A `0`

B `2`

C `4`

D `1`

E None of these.

Question 11: Which of the following statements is **false**?

Both Quicksort *and* Mergesort ...

A are divide-and-conquer algorithms.

B can sort lists of arbitrary length.

C split the problem into two subproblems of similar size.

D can sort lists containing positive as well as negative integers.

E are recursive algorithms.

Question 12: Which of the following expressions is **not** equivalent to the other four?

A `(/) 2`

B `(2/)`

C `\x -> 2 / x`

D `(/2)`

E `\x -> (/) 2 x`

Question 13: Let $f(n) = 3n^2 + 4n + 5$. Which of the following does **not** hold?

A $f(n) = \Omega(n^3)$

B $f(n) = \Omega(n^2)$

C $f(n) = O(n^3)$

D $f(n) = O(n^2)$

E $f(n) = \Theta(n^2)$

Question 14: What is the closed form of the following recurrence?

$$\begin{aligned} T(0) &= 4 \\ T(n) &= T(n-1) + 2n + 3 \quad \text{if } n > 0 \end{aligned}$$

A $T(n) = (n+3)^2$

B $T(n) = n^2$

C $T(n) = (n+2)^2$

D $T(n) = (n+1)^2$

E $T(n) = n \log n^2$

Question 15: Which of the following statements is **false**?

A If $f(x) = \Theta(g(x))$ then $f(x) = O(g(x))$

B $f(x) = \Theta(f(x))$, for all f

C If $f(x) = \Theta(g(x))$ then $g(x) = \Theta(f(x))$

D If $f(x) = O(g(x))$ then $f(x) = \Theta(g(x))$

E If $f(x) = O(g(x))$ then $g(x) = \Omega(f(x))$

Question 16: Use the Master Theorem to find a closed form for the following recurrence:

$$T(n) = 4T(\frac{n}{2}) + n^2$$

The closed form is:

A $T(n) = \Theta(\frac{n^4}{2} + n^2)$

B $T(n) = \Theta(n^2 \log n)$

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

D $T(n) = \Theta(n \log \log n)$

E The Master Theorem does not apply.

Question 17: Which answer is the recurrence best describing the run-time cost of the following Haskell function?

```
foo :: [a] -> Integer
foo [] = 0
foo [x] = 1
foo (x:y:xs) = length (x:xs) + foo xs
```

Assume n is the length of the argument list.

A $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n) = T(n/2) + \Theta(n+1) & \text{if } n > 1 \end{cases}$

B $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n) = 2T(n-2) + 2\Theta(n) & \text{if } n > 1 \end{cases}$

C $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n) = 2T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}$

D $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n) = T(n-1) + \Theta(n-2) & \text{if } n > 1 \end{cases}$

E $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 1 \\ T(n) = T(n-2) + \Theta(n) & \text{if } n > 1 \end{cases}$

Question 18: To implement a complex program, one can use one of three techniques: top-down design, bottom-up design or dodging. Which of the following is **not** true for top-down design?

A Does not break the flow of programming the algorithm.

B Uses existing functions to build new functionality and solve a larger project.

C Relies on building new, simple functions with clearly defined purposes.

D Divides the problem into simpler sub-problems.

E Starts with an overall view of the entire algorithm.

Question 19: In the 8 Step Design Process, how does data description contribute to the overall design of the project?

A Data representation guides the process of dividing the large problem into smaller sub-problems that are easier to solve.

B Data description is very important for large projects, but the programmer can implement small programs without reasoning about data representation.

C Reasoning about data representation helps the programmer to outline the implementation based on inputs and expected outputs.

D Data representation should provide concrete examples of input data, borderline cases, valid and invalid inputs.

E Data representation adds a level of abstraction, helping to separate general ideas from specific implementation decisions.

Question 20: Why are tracing and debugging not enabled by default in every program execution?

A Because it slows down execution.

B Because the programmer can track the errors by reasoning only.

C Because either tracing or debugging is required, but not both.

D Because tracing and debugging cannot be enabled simultaneously.

E Because it would disturb programmers that will maintain the code in the future.