Your exam code:						

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

Teachers: Dave Clarke, Tjark Weber

Bergsbrunnagatan 15, room 1
2014-12-16 / 14:00–19:00

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.
- Read and follow the instructions on the front sheet.
- In the table below, clearly mark **at most one** answer for each question. (If you think that a question is ambiguous or has no correct answer, mark the question number with a $\star$ and explain on a separate sheet of paper what the problem is and what assumptions you have made to answer the question.)
- Tjark Weber will come to the exam hall around 15:30 to answer questions.

Good luck!

Your Answers

Question	Answer					Question	Answer				
1	A	B	C	D	E	11	A	B	C	D	E
2	A	B	C	D	E	12	A	B	C	D	E
3	A	B	C	D	E	13	A	B	C	D	E
4	A	B	C	D	E	14	A	B	C	D	E
5	A	B	C	D	E	15	A	B	C	D	E
6	A	B	C	D	E	16	A	B	C	D	E
7	A	B	C	D	E	17	A	B	C	D	E
8	A	B	C	D	E	18	A	B	C	D	E
9	A	B	C	D	E	19	A	B	C	D	E
10	A	B	C	D	E	20	A	B	C	D	E

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

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 $a \cdot f(n/b) \leq k \cdot f(n)$ for some constant $k < 1$ and all sufficiently large n.

Questions

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

1. Which of the following is a correct (well-typed) Haskell expression?

   (A) `1 ++ 2` (B) `True + 1` (C) `not 0` (D) `length "foo"` (E) `"foo" && "bar"`

   Answer: (D) `length "foo"`

2. What is the type of `head [0<1, 0==1, 0>1]` ?

   (A) The expression is not type-correct. (B) `()` (C) `[Integer]` (D) `Integer` (E) `Bool`

   Answer: (E) `Bool`

3. What is the value of `head [0<1, 0==1, 0>1]` ?

   (A) `[0]` (B) `()` (C) The expression throws an exception. (D) `0` (E) `True`

   Answer: (E) `True`

4. Which of the following is a correct Haskell expression that is equivalent to `3 > 2 || 3 < 1 `div` 0` ?

   (A) `if 3 > 2 then True else 3 < 1 `div` 0`
   (B) `if 3 < 1 `div` 0 then 3 > 2 else False`
   (C) `if 3 > 2 then 3 < 1 `div` 0`
   (D) `if 3 > 2 || 3 < 1 `div` 0 then True`
   (E) `if 3 > 2 then 3 < 1 `div` 0 else False`

   Answer: (A) `if 3 > 2 then True else 3 < 1 `div` 0`

5. Which of the following expressions does **not** have the same value as the other four?

   (A) `['a','b','c']`
   (B) `"a"++"b"++"c"`
   (C) `["abc"]`
   (D) `['a'..'c']`
   (E) `'a':'b':'c':[]`

   Answer: (C) `["abc"]`

6. What is the value of the following expression?

```
let
  x = 1
  f y = let x = y+1 in x+y
  y = f x
in
  x+y
```

   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

   Answer: (D) 4

7. Consider the following function:

```
f x =
  let
    y = 10 `div` x
  in
    if x == 0 then x else y
```

   What is the value of `f 0` ?

   (A) The expression throws an exception. (B) `1` (C) `10` (D) `Infinity` (E) `0`

   Answer: (E) `0`

8. Consider the following function:

```
f (True, 1)  = 1
f (True, _)  = 2
f (_, 1)     = 3
f (False, _) = 4
f (False, 1) = 5
```

   What is the value of `f (False, 0)` ?

   (A) 1 (B) 2 (C) 3 (D) 4 (E) 5

   Answer: (D) 4

9. Which of the following expressions is equivalent to `[1,3..7]` ?

   (A) `[1,3,5,7]` (B) `[1,2,3,4,5,6,7]` (C) `[1,4,7]` (D) `[]` (E) `[1,3,4,5,6,7]`

   Answer: (A) `[1,3,5,7]`

10. Recall the mergesort algorithm. To ensure that the output list is a sorted permutation of the input list, it is important that the splitting function ...

    (A) returns two lists that together contain the same elements as the input list.
    (B) returns two sorted lists.
    (C) splits the input list into elements below the pivot, and elements above the pivot.
    (D) returns two lists of equal length.
    (E) runs in $\Theta(n)$.

    Answer: (A)

11. Which of the following best describes the purpose of data descriptions in the 8 step design process?

    (A) To informally describe the input and output of functions.
    (B) To give examples of the data used in a program.
    (C) To describe how the data in the program relates to the real world concepts it models and give constraints on the validity of the data.
    (D) To give examples of valid and invalid data for the program.
    (E) To provide a description of the relationship between data used in programs and the functions that operate on that data.

    Answer: (C)

12. What of the following is **not** an accurate description of the goal of stepwise refinement?

    (A) To help break a large problem into smaller problems.
    (B) To solve a problem by introducing other functions to help solve the problem.
    (C) To break a function into details which are refined in successive steps until the whole program is fully defined.
    (D) To refine the steps of an algorithm into lines of code.
    (E) To refine a highly abstract representation of some required program gradually through a sequence of intermediate representations to yield a final program in some chosen programming language.

    Answer: (D)

13. Which of the following post-conditions is best for the given code?

```
rempos :: [Integer] -> [Integer]
rempos []                 = []
rempos (a:as) | a >= 0    = a : rempos as
rempos (_:as) | otherwise = rempos as
```

    (A) The length of the result is greater than or equal to zero.
    (B) The input list traversed recursively to remove all negative numbers.
    (C) A list of integers with negative elements removed.
    (D) The input list with negative elements removed.
    (E) The input list with just positive elements remaining.

    Answer: (D). Answer (C) is ruled out because it states the result type (and is imprecise), which is already specified and thus superfluous.

14. Which of the following is a variant for the function `rempos` from the previous question?

(A) the length of the input list (B) `[]` (C) `as` (D) `a:as` (E) 1

Answer: (A) the length of the input list

15. Let $f(n) = 4n^3 + 99n^2 + 3$. Which of the following is **not** correct?

(A) $f(n) = O(n^2)$ (B) $f(n) = \Theta(n^3)$ (C) $f(n) = O(n^4)$
(D) $f(n) = \Omega(n^2)$ (E) $f(n) = \Omega(n^3)$

Answer: (A) $f(n) = O(n^2)$

16. What is the *most precise* closed form for the following recurrence?

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

(A) $T(n) = O(n)$ (B) $T(n) = O(n^4)$ (C) $T(n) = \Omega(n^2)$
(D) $T(n) = \Theta(n^2)$ (E) $T(n) = \Theta(n^3)$

Answer: (E). All but (A) and (D) are valid, but (E) is most precise.

17. Use the Master Theorem to find a closed form for the following recurrence:

$$T(n) = 2T(n/2) + n \log n$$

The closed form is:

(A) $\Theta(n)$
(B) $\Theta(n \log^2 n)$
(C) $\Theta(n \log n)$
(D) $\Theta(n^2)$
(E) The Master Theorem does not apply.

Answer: (B) $T(n) = 2T(n/2) + n \log n$ implies $T(n) = \Theta(n \log^2 n)$ (Case 2)

18. Which answer is the recurrence representing the run-time cost of the following function?

```
f :: Integer -> Integer
f n | n <= 1 = 1
f n          = f (n - n `div` 2) + f (n - n `div` 2)
```

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

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

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

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

(E) $T(n) = \begin{cases} \Theta(1) & \text{if } n \leq 0 \\ 2T(n/2) & \text{otherwise} \end{cases}$

Answer: (C)

19. What is the type of the following function?

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

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

Answer: (B) `[a] -> [a]`

20. Suppose you want to write a function `increment_even :: [Integer] -> [Integer]` that returns its input list with all odd numbers removed, and all even numbers incremented by `1`. For instance, `increment_even [1,2,5,7,8] == [3,9]`. Which of the following function definitions can you **not** use?

(A)
```
increment_even xs = [ x+1 | x <- xs, even x ]
```

(B)
```
increment_even [] = []
increment_even (x:xs) | even x = (x+1) : increment_even xs
increment_even (_:xs)          = increment_even xs
```

(C)
```
increment_even = map (+1) . filter even
```

(D)
```
increment_even xs = map ((+1) . filter even) xs
```

(E)
```
increment_even xs = map (+1) (filter even xs)
```

Answer: (D) `increment_even xs = map ((+1) . filter even) xs`