Final Exam (Part 2) in
Program Design and Data Structures (1DL201)

Teachers: Dave Clarke, Tjark Weber

Bergsbrunnagatan 15
2016-03-17 / 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.
- Dave Clarke will come to the exam hall around 9:30 to answer questions.

Good luck!

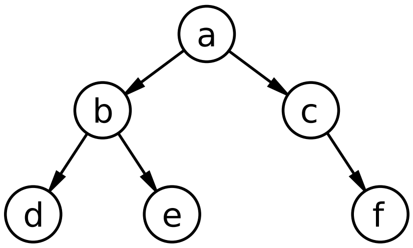
Questions

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

Question 1: Suppose you want to write a physics simulation, where objects move through space. Each object has a speed, given by a value of type `Double`. Speeds are measured either in meters per second (m/s) or in kilometers per hour (km/h). Which of the following type declarations could you use to make sure that speeds measured in m/s are not confused with speeds measured in km/h?

- □ A `type Speed = Double`
- □ B `type SpeedMs = Double; type SpeedKmh = Double`
- □ C `data Speed = Ms | Kmh`
- □ D `data Speed = Speed Double`
- □ E `data Speed = Ms Double | Kmh Double`

Question 2: In what order would a post-order traversal process the values in the following binary tree?



□ A abdecf □ B abcdef □ C dbeacf □ D debfca □ E fedcba

Question 3: Consider the type of general trees, defined as

```
data Tree a = Node a [Tree a]
```

What does the following function (`f`) compute?

```
f (Node a ts) = 1 + fs ts
  where
    fs [] = 0
    fs (t:ts) = f t + fs ts
```

- □ A The sum of all values in a tree.
- □ B The number of nodes in a tree.
- □ C The height of a tree.
- □ D A post-order list of all values in a tree.
- □ E A pre-order list of all values in a tree.

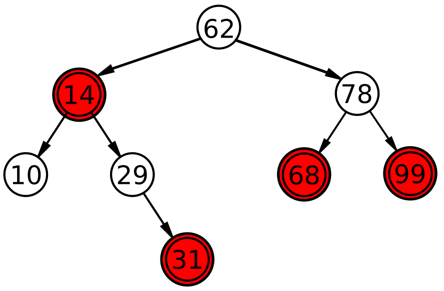
Question 4: The (worst-case) complexity of inserting an item into a binary search tree with n nodes is

A $O(2^n)$ B $O(n)$ C $O(1)$ D $O(n^2)$ E $O(\log n)$

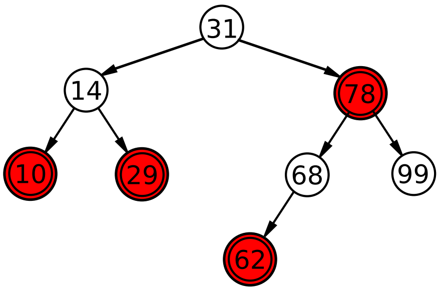
Question 5: Suppose the following numbers are inserted, in the given order, into an initially empty red-black tree. (Grey=red.)

29 62 78 10 14 99 31 68

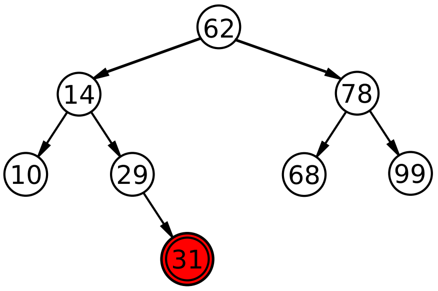
What is the resulting red-black tree?



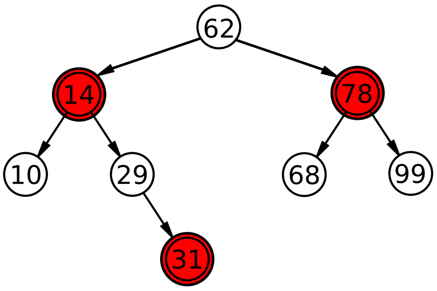
A



C



B



D

E None of these.

Question 6: How many nodes does a binomial tree of rank r have?

A r B $\lfloor \lg r \rfloor + 1$ C $\binom{r}{k}$ D 2^r E 1

Question 7: Which of the following is a distinguishing feature of *abstract data types*? (**distinguishing**: characteristic of one thing or person, so serving to identify it; distinctive.)

A Pattern matching.

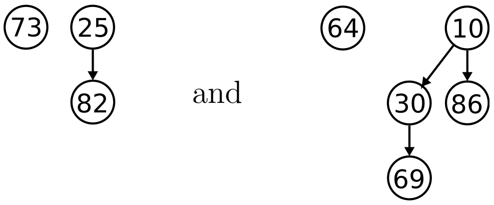
B Hidden implementation details.

C More efficient than other kinds of data types.

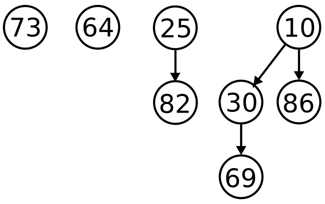
D Polymorphism.

E Different constructors representing different kinds of data.

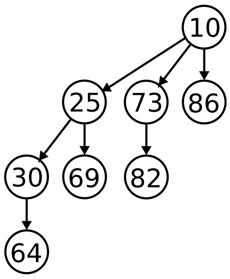
Question 8: What is the result of merging the following two binomial heaps?



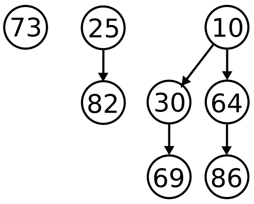
A



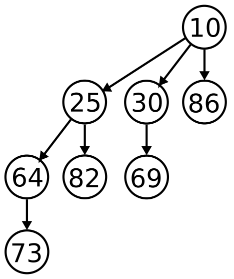
B



C



D



E None of these.

Question 9: Which of the following best describes the concept of *overloading*?

A Two or more functions with the same name but different definitions.

B Two or more functions with different definitions.

C Two or more functions with the same definition.

D Computationally-intensive functions that overload the CPU.

E Polymorphic functions.

Question 10: Which of the following, when executed from within a Haskell program, is **not** a side-effect?

A Writing *one does not simply walk into Mordor* to the screen.

B Halting the program with an `EverWatchfulEye` error.

C Playing Soundgarden's *Spoonman* through the speakers.

D Recursively calculating the number of times the phrase *there is evil there that does not sleep* occurs in a string.

E Reading Tolkien's *The Lord of the Rings* from a text file.

Question 11:
Consider the type class `Mirror` which has a function `mirror ::  t -> t` whose intention is to produce some kind of mirror image of a data structure.

```
class Mirror t where
    mirror :: t -> t

instance Mirror () where
    mirror () = ()

instance Mirror Bool where
    mirror = not
instance Mirror Int where
    mirror n = -n

instance (Mirror a, Mirror b) => Mirror (a,b) where
    mirror (a,b) = (mirror a, mirror b)

data BTree a b = Leaf a | Branch b (BTree a b) (BTree a b)

instance (Mirror a, Mirror b) => Mirror (BTree a b) where
    -- CODE MISSING

instance Mirror a => Mirror [a] where
    mirror = reverse . map mirror
```

Given the test case

```
mirror $ Branch (True,()) (Leaf [1,2,3])
                          (Branch (False,()) (Leaf [3,4,5]) (Leaf [5,6,7]))
     = Branch (False,())
              (Branch (True,()) (Leaf [-7,-6,-5]) (Leaf [-5,-4,-3]))
              (Leaf [-3,-2,-1])
```

which of the following is the most reasonable implementation of `mirror` for `BTree`?

☐ A
```
mirror (Leaf a) = mirror (Leaf a)
mirror (Branch b l r) = Branch (mirror b) (mirror r) (mirror l)
```

☐ B
```
mirror (Leaf a) = Leaf (mirror a)
mirror (Branch b l r) = Branch (mirror b) (mirror r) (mirror l)
```

☐ C
```
mirror (Leaf a) = Leaf (mirror a)
mirror (Branch b l r) = Branch b (mirror l) (mirror r)
```

☐ D
```
mirror (Leaf a) = Leaf a
mirror (Branch b l r) = Branch b (mirror l) (mirror r)
```

☐ E
```
mirror (Leaf a) = Leaf (mirror a)
mirror (Branch b l r) = Branch b (mirror r) (mirror l)
```

Question 12: Which of the following is true about native/truly mutable arrays (the ones in `Data.Array.IO`) in Haskell?

A $O(1)$-time read, $O(1)$-time write.

B $O(\log n)$-time read, $O(1)$-time write.

C $O(\log n)$-time read, $O(\log n)$-time write.

D $O(1)$-time read, $O(\log n)$-time write.

E $O(1)$-time read, $O(n)$-time write.

Question 13: Consider the following code:

```
main = do
  putStrLn $ "13"
  let x = putStrLn "10"
  y <- putStr "12"
  x
  putStrLn "11"
  return y
```

What is the output (not the result) of evaluating `main` in `ghci`?

A
```
13
1210
11
12
```

B
```
13
12
10
11
```

C
```
13
10
12
11
```

D
```
13
1210
11
```

E
```
13
10
1211
12
```

Question 14: Assume that a stack contains the entries

(top) | 12 | 15 | 12 | 16 | 12 | 17 | (bottom).

Which stack is the result of performing the following operations

`pop`, `pop`, `push` 12, `pop`, `pop`, `push` 12 ?

A | 12 | 15 | 12 | 12 |

B | 12 | 17 | 12 | 12 |

C | 12 | 12 | 12 | 15 |

D | 12 | 12 | 12 | 12 |

E | 12 | 16 | 12 | 17 |

Question 15: Given a hashtable with 7 slots that contain, respectively, chains of length 1, 2, 0, 0, 3, 1, 0. That is, slot 0 contains a chain of length 1, slot 1 contains a chain of length 2, slot 3 is empty, etc. Keys are numbers and the hash function is $hash(key) = key \bmod 7$.
What is the length of the longest chain after inserting the numbers 12, 16, 34, 7, 15, 21, 22, assuming that *none* of these keys already appear in the table?

A 3 B 4 C 5 D 6 E 7

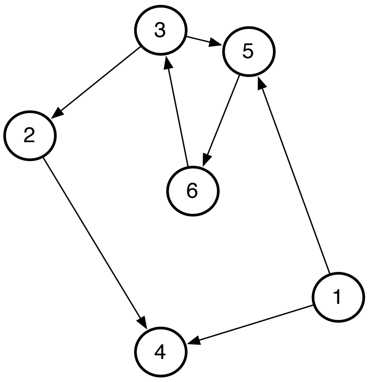
Question 16:
Consider the following hash table of 11 cells, where $\perp$ denotes that a cell was never used.

0	1	2	3	4	5	6	7	8	9	10
33	23	57	35	70	$\perp$	6	$\perp$	19	$\perp$	54

Assume that the hash function is $h(k) = k \bmod 11$, that open addressing with linear probing function $f(i) = i$ is used as the conflict resolution method, and that duplicates are allowed.
Firstly, 35 is deleted from the hash table. In which cell of the resulting table will 89 be placed?

$\boxed{\text{A}}$ Nowhere $\boxed{\text{B}}$ 3 $\boxed{\text{C}}$ 5 $\boxed{\text{D}}$ 9 $\boxed{\text{E}}$ 10

Question 17: Consider the following graph:



Which of the following is a valid adjacency list representation of the graph?

$\boxed{\text{A}}$

	1	2	3	4	5	6
1				1	1	
2			1	1		
3		1			1	1
4	1	1				
5	1		1			1
6			1		1	

$\boxed{\text{B}}$

	1	2	3	4	5	6
1				1	1	
2				1		
3		1			1	
4						
5						1
6			1			

$\boxed{\text{C}}$

1	4, 5
2	3, 4
3	2, 5, 6
4	1, 2
5	1, 3, 6
6	3, 5

$\boxed{\text{D}}$ (1,4), (1,5), (2,4), (3,2), (3,5), (5,6), (6,3)

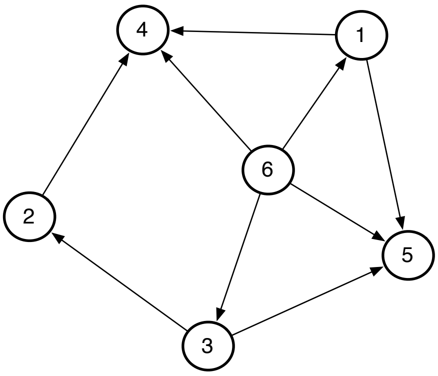
$\boxed{\text{E}}$

1	4, 5
2	4
3	2, 5
4	
5	6
6	3

Question 18: Consider the graph from the previous question. Which of the following is a valid depth-first search ordering—that is, the order in which nodes are visited when performing a depth-first search?

□A 154326 □B 156342 □C 324561 □D 623451 □E 635214

Question 19: Consider the following graph:



Which of the following is a valid topological sort of this graph?

□A 123456 □B 654321 □C 243615 □D 516342 □E 613245

Question 20: Consider the following code that uses exceptions, where `return :: a -> Exceptional a` and `throw :: Exception -> Exceptional a` are used to report good and bad values, respectively.

```
data Exception = DivideByZeroException | BadWordException
               | ConquerByZorroException

type Exceptional a = Either Exception a

(///) :: Int -> Int -> Exceptional Int
_ /// 0 = throw DivideByZeroException
a /// b = return $ a `div` b

cleanSentence :: String -> Bool
cleanSentence s = and $ (map clean) (words s)
  where clean s = not (s `elem` ["flip", "jeez", "crumbs"])

censor :: String -> Exceptional String
censor s = if cleanSentence s then return s else throw BadWordException

duplicate :: Int -> String -> Exceptional String
duplicate n s | n < 0 = throw ConquerByZorroException
duplicate n s | otherwise = return $ concat $ replicate n s

prog :: Int -> Int -> String -> Exceptional String
prog a b s = do
  e <- a /// b
  f <- censor s
  duplicate e f
```

Under what conditions on inputs `a`, `b`, and `s` does function `prog` result in a `ConquerByZorroException`?

A Whenever `a` is negative and `s` contains no bad words.

B Whenever `b` is not zero, ``a `div` b`` is negative, and `s` contains no bad words.

C Whenever `b` is not zero, ``a `mod` b`` is negative, and `s` contains no bad words.

D Whenever neither `a` nor `b` is zero, at most one of `a` or `b` is negative, and `s` contains no bad words.

E Whenever neither `a` nor `b` is zero, at most one of `a` or `b` is negative, and `s` is not empty and contains no bad words.

*Do **not** write above this line!*

Answer Sheet — Exam 1DL201 of 2016-03-17

Instructions: Using a **dark** colored pen, fill in **at most one** answer box (A to E) per question. Fill the answer box **entirely** (■)—we will use an optical character recognition (OCR) system that may not recognize ticks, crosses, circles, etc.

If you think that a question is ambiguous or has no correct answer, mark the question number with a ⋆ and explain **on the backside of this sheet** what the problem is and what assumptions you have made to answer the question.

Transfer your answers from the question sheets to this answer sheet **just before handing in**. If you want to change an answer, then please request a new answer sheet. You may keep the question sheets; at the end of the exam, **only hand in this answer sheet**.

Also fill in your **exam code** in clear handwriting at the bottom of this page.

Grading:

Correct answers	≤ 9	10-13	14-16	17-20
Grade	U	3	4	5

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

Again: Please fill your chosen boxes **entirely** and in **dark** colored pen!

Your exam code: □□□□□□