Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion

Sandra Dylus, Priv.-Doz. Dr. Frank Huch, Niels Bunkenburg

3. Klausur zur Vorlesung „Advanced Programming (pre-masters)“

WS 18/19

You can obtain 28 points within two assignments. To pass the test, you have to reach at least 14 points.

You have 90 minutes to complete the test. It is not allowed to use any material other than a pen. Electronic devices have to be turned off.

Hold your student card nearby, we will check it during the examination.

You will be informed about the results on Wednesday, January 23rd (9:00 to 10:00 AM) in room number 715 in CAP 4.

Aufgabe 1 - Multiple Choice and Questions 12 Punkte

Answer the following questions directly on this sheet of paper.

For the first five questions, simply mark all correct answers with an X. **If you want to change your answer after marking a statement, fill the original square and draw a new one, as shown in the example.** Note that any number of answers can be correct. Each question is worth 1.5 points. For each incorrect answer 0.5 points are deducted; negative scores are not possible.

Example

1. Which names are associated with Advanced Programming?

   a) ☒ Sandra
   b) ☐ Peter
   c) ☒ Frank
   d) ■ ☐ Rebecca
   e) ☒ Niels

Multiple Choice

1. Which of the following expressions yield `[1,2,3,4,5]`?

   a) ☐ `[ x | x <- [1,2,3,4,5]]`
   b) ☐ `[ even x | x <- [1,2,3,4,5]]`
   c) ☐ `[ x / 2 | x <- [1..10], even x]`
   d) ☐ `[ x - y | x <- [1,2,3,4,5], y <- [0,0,0,0,0]]`
   e) ☐ `[ y | x <- [[1],[2,3],[4],[5]], y <- x]]`

2. Which of the following functions are part of the type class `Ord`?

   a) ☐ `(==)`
   b) ☐ `(<=)`
   c) ☐ `ordering`
   d) ☐ `compare`
   e) ☐ `lt`

3. Which of the following expressions terminate (if they are entered into ghci)?

   a) ☐ `zip [1..] [2,4,6,8,10]`
   b) ☐ `take 10 [10..]`
   c) ☐ `let noLoop = noLoop in head noLoop`
   d) ☐ `let xs = 1 : xs in head xs`
   e) ☐ `let xs = 1 : xs in tail xs`

4. Which of the following expressions are equivalent to `getLine >>= \str -> return (reverse str)`?

   a) ☐ `getLine >> return (reverse str)`
   b) ☐ `getLine`
   c) ☐ `getLine >>= \str -> return (reverse str) >> return 42`
   d) ☐ `getLine >>= return . reverse`
   e) ☐ `getLine >>= \str -> reverse str`

5. Which of the following types are correct?

   a) ☐ `getChar :: Char -> IO ()`
   b) ☐ `putStrLn :: String -> IO String`
   c) ☐ `getLine :: IO String`
   d) ☐ `putStr :: String -> IO ()`
   e) ☐ `print :: Show a => a -> String`

Questions

1. (2P) Translate the following IO program with do-notation into an equivalent programm without do.

```
main = do
  x <- getLine
  y <- return "Hello"
  putStrLn (x ++ y)
```

2. (1 P) Give types for the following IO functions from the Prelude.

```
(>>=) ::
```

```
return ::
```

3. (1.5 P) Describe the behaviour of the following program with at most two sentences. You can assume that the user enters a positive natural number.

```
main = do
  str <- getLine
  print ([(read str)::Integer ..])
```

Aufgabe 2 - I/O, Type Classes and Lazy Evaluation 16 Punkte

1. (10P) Implement a game where a user needs to guess a secret number. An exemplary round looks as follows.

   ```
   *Main> main
   Your guess, please!
   hello?
   Enter a number, please.
   13
   Too small
   Your guess, please!
   73
   Too large
   Your guess, please!
   42
   Congrats, your guess is correct!
   You needed 3 tries to win the game!
   ```

   The lines “hello?”, “13”, “73” and “42” were entered by the user.

   As a first step, define a helper function `getInt :: IO Int` that handles non-digit input like we see above for “hello?” until the user enters a positive number. You can use `all isDigit :: String -> Bool` to check if a string contains only digits.

   Define the function `guessNumber` that is used in the `main` function as defined below. Your implementation of `guessNumber` should use the helper function `getInt` and behave as shown in the example.

   ```haskell
   main :: IO ()
   main = do
     n <- guessNumber 42
     putStrLn ("You needed " ++ show n ++ " tries to win the game!")
   ```

2. (6P) We consider the following algebraic data type for leaf-labeled trees.

   ```haskell
   data T a = L a | B (T a) (T a)
   ```

   - Implement an instance for the type class `Eq`. The implementation should behave similar to the derived instance.
   - Define an infinite tree `infTree :: T ()`.