Exam
Mark each answer with your initials. Write clearly and comment what you do, that might give you points even if the result is wrong. Each question is worth five points.

1. Define the following function with pattern matching:

   ```haskell
   test :: Bool -> Bool -> Bool -> Bool
   test a b c
   | a and b = not c
   | b and c = not a
   | a and c = not b
   | otherwise = False
   ```

2. Rewrite the definition of \( g \) so that the argument \( x \) no longer appear on the left hand side of the equation.

   \[
g f x = f ((f x)/3)
\]

3. What is the type of \( e \) defined below? The answer should include a motivation.

   ```haskell
   e k = do
   x <- k
   return (2*x)
   return False
   ```

4. What is the type of \( \text{col} \) defined below? The answer should include a motivation.

   ```haskell
   col = white `switch` ((key `snapshot` col) =>> \(c,old\) ->
   case c of 'R' -> red
   'B' -> blue
   'Y' -> yellow
   _   -> lift0 old)
   where:
   white, red, blue, yellow :: Behavior Color
   key       :: Event Char
   lift0     :: a -> Behaviour a
   (=>>)     :: Event a -> (a -> b) -> Event b
   switch    :: Behavior a -> Event (Behavior a) -> Behavior a
   snapshot  :: Event a -> Behavior b -> Event (a, b)
   ```

5. What is a monad? What are the benefits of this concept?

6. The list library contains the following definition:

   ```haskell
   unfoldr :: (b -> Maybe (a,b)) -> b -> [a]
   unfoldr f b = case f b of
      Nothing -> []
      Just (a,b) -> a : unfoldr f b
   ```

   Under what conditions is the following true?

   \[
   \text{unfoldr } f' \ (\text{foldr } f \ z \ \text{xs}) = \text{xs}
   \]