CS 421 --- Macros Activity

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to class	
Reflector	Assesses team performance	

Purpose

You have seen macros in the Scheme MP. We will talk about a few ways of using them in Haskell.

Part 1 --- Writing Code using Print

Consider the following code session.

```
*Main> putStrLn s
hi
*Main> print s
"hi"
*Main> putStrLn s >> print s
hi
"hi"
```

Problem 1) What is the difference between `print` and `putStrLn`?

Now consider this Haskell file `Q.hs`. [1]

```
main = putStr s >> print s
s = "main = putStr s >> print s\ns="
```

Problem 2) What does it do?

[1]Credit: Daniel Martin at http://www.iis.sinica.edu.tw/ scm/2007/a-haskell-quine/

Part 2 --- Compiling an AST to Strings

Problem 3) Consider the following data type:

```
data Ops = Val Integer
         | Times
         | Plus
  deriving Show
```

Problem 4) Write a Haskell function `compile :: [Ops] -> String` that takes a list of `Ops` and compiles it into a Haskell expression.

```
Prelude> compile [Val 10, Val 20, Plus, Val 30, Val 40, Plus, Times]
"((10 + 20) * (30 + 40))"
```

Part 3 --- Template Haskell

You can use the `-XTemplateHaskell` extension to get macros in Haskell!
Problem 5) Consider the following code sample.

```
-> % stack ghci
Prelude> :set -XTemplateHaskell
Prelude> import Language.Haskell.TH
Prelude> id = [| \x -> x + 1 |]
Prelude> runQ id
LamE [VarP x_0] (InfixE (Just (VarE x_0)) (VarE GHC.Num.+) (Just (LitE (IntegerL 1))))
Prelude> $(id) 10
11
Prelude> let mkInc delta = [| \x -> x + delta |]
Prelude> let five = $(mkInc 5)
Prelude> five 10
15
Prelude> runQ [| \x y z -> z |]
LamE [VarP x_1,VarP y_2,VarP z_3] (VarE z_3)
Prelude> runQ [| \f z -> f z |]
LamE [VarP f_4,VarP z_5] (AppE (VarE f_4) (VarE z_5))
```

Problem 6) What is the purpose of `[| |]` and `$()` brackets?

Problem 7) What are all the `VarP` and related types doing?

Consider the following two files.
File 1: `TH.hs`

```
{-# LANGUAGE TemplateHaskell #-}

module TH where

import Language.Haskell.TH

mkInc :: String -> Integer -> Q [Dec]
mkInc name delta =
  do body <- [| \x -> x + delta |]
     let n = mkName name
     return $ [FunD n [Clause [] (NormalB body) []]]
```

File 2: `foo.hs`

```
{-# LANGUAGE TemplateHaskell #-}

import TH

$(mkInc "five" 5)
```

Problem 8) What will this code do?

Now consider this code. What does it do?
File 1: `TH.hs` (partial)

```haskell
mkMapTuple :: Integer -> Q [Dec]
mkMapTuple size = do
  vars <- mapM newName (map (\n -> "x" ++ show n) [1..size])
  let args = map VarP vars
  f <- newName "f"
  let fargs = map (\v -> AppE (VarE f) (VarE v)) vars
  let body = LamE [VarP f, TupP args] (TupE fargs)
  return $ [FunD (mkName $ "mapTup" ++ show size)
           [Clause [] (NormalB body) []]]
```

File 1: `foo.hs`

```haskell
$(mkMapTuple 3)
```

Problem 9) What is the difference between `mkName` and `makeName`? Hint: notice that `newName` is monadic, but `mkName` is not.

Problem 10) Try to write `mkRevTuple` that creates a tuple reverser of a given size.

Macros Activity --- Team's Assessment (SII)

Manager or Reflector: Consider the objectives of this activity and your team's experience with it, and then answer the following questions after consulting with your team.

1. What was a **strength** of this activity? List one aspect that helped it achieve its purpose.

2. What is one things we could do to **improve** this activity to make it more effective?

3. What **insights** did you have about the activity, either the content or at the meta level?

Macros Activity--- Reflector's Report

Manager	Keeps team on track	
Recorder	Records decisions	
Reporter	Reports to Class	
Reflector	Assesses team performance	

1. What was a strength of your team's performance for this activity?

2. What could you do next time to increase your team's performance?

3. What insights did you have about the activity or your team's interaction today?