Major Assignment 2

Date of Submission - 18th Oct 2013 Weightage - 20%

This is the second of the three parts of a major assignment on the implementation of a Haskell-like language, called μ-Haskell. The three parts are:

1. Writing a parser to convert programs in μ-Haskell to an internal representation in Haskell.
2. Translating the internal representation to G-code.
3. Interpreting G-code.

We shall provide you a model parser for the language that will convert a μ-Haskell program into this internal representation. Here are the datatypes for the internal representation:

```
type Fname = String
type Var = String
data Program = Prog [Fundef] Exp deriving Show
data Fundef = Fun String [String] Exp deriving Show
data Exp =  I Int | V Var | B Bool | Nil |
            Fname String |   --  both user defined and builtin
            App Exp Exp deriving Show
```

Here are some example programs converted into μ-Haskell.

```
program1 = (2 + 3)
program1 = Prog [] (App (App (Fname "+") (I 2)) (I 3))

program2 = f x y = x + y
           f 2 3
program2 = Prog
           [Fun "f" ["x", "y"] (App (App (Fname "+") (V "x")) (V "y"))]
           (App (App (Fname "f") (I 2)) (I 3))

program3 = fact n = if n == 0 then  else n * fact (n - 1)
program3 = Prog
           [Fun "fact" ["n"]
                  (App (App (App (Fname "if")
                                 (App (App (Fname "==") (V "n")) (I 0)))
                            (I 1))
```

```
                        (App (App (Fname "*") (V "n"))
                             (App (Fname "fact")
                                  (App (App (Fname "-") (V "n")) (I 1)))))]
           (App (Fname "fact") (I 5))
```

What you have to do is to write a function called `gencpgm` which will take a program in this representation and convert it into a list of instructions.

```
type Code = [Instn]
gencpgm :: Program -> Code
```

Here is the datatype representing instructions and a show function for it.

```
data Instn = PUSH Int | PUSHINT Int | PUSHGLOBAL String |
             PUSHBOOL Bool | PUSHNIL | POP Int |
             EVAL | UNWIND | MKAP | UPDATE Int |  RETURN |
             LABEL String | JUMP String | JFALSE String |
             ADD | SUB | MUL | CONS | HEAD | TAIL | IF | EQU |
             GLOBSTART String Int | PRINT | STOP

instance Show Instn where
  show (PUSH i) = "   PUSH " ++ show i ++ "\n"
  show (PUSHINT i) = "   PUSHINT " ++ show i ++ "\n"
  show (PUSHGLOBAL str) = "   PUSHGLOBAL " ++ show str ++ "\n"
  show (PUSHBOOL b) = "   PUSHBOOL " ++ show b ++ "\n"
  show PUSHNIL  = "   PUSHNIL " ++ "\n"
  show (POP i) = "   POP " ++ show i ++ "\n"
  show EVAL = "   EVAL" ++ "\n"
  show UNWIND = "   UNWIND" ++ "\n"
  show MKAP = "   MKAP" ++ "\n"
  show RETURN = "   RETURN" ++  "\n"
  show (UPDATE i) = "   UPDATE " ++ show i ++ "\n"
  show (LABEL str) = "LABEL " ++ show str ++ "\n"
  show (JUMP str) = "   JUMP " ++ show str ++ "\n"
  show (JFALSE str) = "   JFALSE " ++ show str ++ "\n"
  show ADD = "   ADD" ++ "\n"
  show SUB = "   SUB" ++ "\n"
  show MUL = "   MUL" ++ "\n"
  show CONS = "   CONS" ++ "\n"
  show HEAD = "   HEAD" ++ "\n"
  show TAIL = "   TAIL" ++ "\n"
  show IF = "   IF" ++ "\n"
  show EQU = "   EQU" ++ "\n"
  show (GLOBSTART str i) = "\n GLOBSTART " ++ show str ++ " " ++
                           show i ++ "\n"
```

```
show PRINT = "   PRINT" ++ "\n"
show STOP = "   STOP" ++ "\n"
```