Category "intro 2017/2018"

Q1 lisp - fix bug (10 pts)

After completing task **1.2.(a)**, enter the correct version of line 2 in the box below.

answer:

```
(print (+ 1 2))
```

comment:

In Lisp arithmetic operators are treated as functions and has to appear at the front before the two operands.

Q2 lisp - if (15 pts)

After completing task **1.2.(b)**, enter the correct version of line 8 in the box below.

answer:

```
(if (string= input "")
    (format t "nothing typed...")
    (format t "you typed: ~a" input))
```

comment:

Note that this "if" is a function with 3 parameters. These 3 parameters are analogous to the 3 parameters of Java's `?:` *operator, ie condition, then-branch and else-branch.*

Q3 lisp - eval (20 pts)

After completing task **1.2.(c)**, enter the correct version of line 11 in the box below.

answer:

```
(let ((prg '(+ 1 n))) (progn (print prg) (setf n 1) (print (eval prg))))
;; OR:
(let ((prg '(+ 1 n))) (print prg) (setf n 1) (print (eval prg)))
```

comment:

Note that `(progn ...)` *can be omitted as* `(let ...)` *can be also used to give a sequence of expressions. In any case, the values of expressions* `(print prg)` `(setf n 1)` *are ignored. These two expressions exist only for their side effects, ie* `(print prg)` *prints to console and* `(setf n 1)` *sets a global variable. This code illustrates how Lisp is both a functional and imperative programming language at the same time. Haskell separates functional and imperative programming more thoroughly. Due to such side effects, in Lisp it is important to know the evaluation order. In Haskell the order of evaluating expressions is irrelevant. The Haskell compiler is thus allowed to choose the evaluation order and optimise it.*

Q4 haskell - ask, repeat prompt (10 pts)

After completing task **1.3.(b).(ii)**, enter the updated code extract in the box below.

answer:

```
ask prompt =
  do
  putStrLn prompt
  line <- getLine
  if line == ""
    then ask "try again" -- the only change
    else putStrLn ("you said: " ++ reverse line)
```

Q5 haskell - ask, append prompt (10 pts)

After completing task **1.3.(b).(iii)**, enter the updated code extract in the box below.

answer:

```
ask prompt =
  do
  putStrLn prompt
  line <- getLine
  if line == ""
    then ask (prompt ++ "!") -- the only change
    else putStrLn ("you said: " ++ reverse line)
```

Q6 haskell - explain error (15 pts)

After completing task **1.3.(b).(iv)**, explain the error message in your own words, using plain English.

answer:

The function `ask` *takes one parameter of type* `String`*. This code change violates this because the parameter passed to the function is not a string but an imperative program (ie type* `IO ()`*)*

Q7 haskell - compare with Lisp (20 pts)

Answer the question stated in task **1.3.(b).(v)**.

answer:

The Lisp analogue of `putStrLn` *would not only print the given string but also return this string as a result. Thus the code will work as before, except prompt will be printed twice instead of once. In particular, unlike Haskell, Lisp allows one to freely mix functions that have side effects with functions that return values. Lisp* `print` *returns a value and performs an imperative action as well.*