CMPT 383
Quiz #7
November 24, 2005

1) Write all reduction sequences for the expression `square(5+2)`.

```
square          :: Integer → Integer
square x        = x * x
   square (5+2)             square (5+2)             square(5+2)
   {def +}                  {def square}             {def square}
   square 7                 (5+2) * (5+2)            (5+2) * (5+2)
   {def square}             {def + on first}         {def + on second}
   7 * 7                    7 * (5+2)                (5+2) * 7
   {def *}                  {def +}                  {def +}
   49                       7 * 7                    7 * 7
                            {def *}                  {def *}
                            49                       49
```

2) Using the function `square`, design a function `quad` that raises its argument to the fourth power.

```
quad   :: Integer → Integer
quad x = square (square x)
```

3) Define a function for computing the area of a circle with given radius `r` (declare `pi` as a `float` with value `22/7`).

```
square :: Float → Float
square x = x * x

pi :: Float
pi = 22/7

area_circle   :: Float → Float
area_circle r = pi * square r
```

4) Define a function `abs` that returns the absolute value of an integer.

```
abs :: Integer → Integer
abs x = if x < 0 then –x else x
```

5) Put the following strings in ascending order: "`McMillan`", "`Macmillan`", and "`MacMillan`".

```
"MacMillan" < "Macmillan" < "McMillan"
```

6) What are the values of the following expressions?

a) `show(show 50)`
   "50"

b) `show 50 ++ show 50`
   "5050"

c) `putStr("Results= " ++ show(3*33) ++ " and " ++ square(10))`
   It will generate an error because the output of the function square is a numeric value and the function ++ takes two lists as input.
   ```
   putStr("Results= " ++ show(3*33) ++ " and " ++ show(square(10)))
   generates
   Results= 99 and 100
   ```

7) Suppose we curry the arguments of the function `delta`, so that we can write `delta a b c` rather than `delta(a,b,c)`. What is the type of the curried version?

```
delta :: (Float,Float,Float) → Float
```

`delta :: Float → Float → Float → Float`

8) Suppose we define multiply by

```
multiply          :: (Integer,Integer) → Integer
multiply(x,y)     = if x==0 then 0 else x*y

infinity    :: Integer
infinity    = infinity + 1
```

The symbol == is used for an equality test between two integer. Assume that the evaluation of `e1==e2` proceeds by reducing `e1` and `e2` to canonical form and testing whether the two results are identical.
Under lazy evaluation:

a) What would be the value of `multiply(0,infinity)`?
0

b) What would be the value of `multiply(infinity,0)`?
^