CS 457/557 Homework 2 – due 2pm, Tuesday, October 11, 2005

Hand in your solutions on paper *and* email them to `cs457acc@cs.pdx.edu`. All the programs should be placed in a single file `hw2.hs`, which should be sent as an *attachment* to your email message. It is *not* necessary to show evidence that you have loaded and tested your programs, but this is of course the only sensible way to make sure that you have found correct answers!

1. Define a function

```
same :: IO Bool
```

that (i) reads two lines from standard input; (ii) if the two lines are identical, prints "`yes`" and otherwise prints "`no`"; (iii) returns the value `True` iff the two lines are identical.

2. Define a function

```
average :: IO ()
```

that reads a number (call it n) from standard input, then reads a further n numbers, and finally prints out their average on standard output. You can assume that each number appears on a separate line of the input. You can convert strings to numbers using the `read` function from the Prelude. Try to keep the core functionality (computing the average) separate from the IO processing as much as possible.

3. Define a function

```
while :: IO Bool -> IO () -> IO ()
```

such that `while` *test* *oper* performs *oper* while *test* equals `return True`. Here's an example of how this function might be used (using the `same` function from exercise 1):

```
while same
      (putStrLn " - not different yet")
```

4. Write a function

```
me :: IO ()
```

that uses the graphics functions from Ch. 3 to create a large window and draw a big representation of the first letter of your name in it. The window should stay visible until the space key is pressed while the keyboard focus is in the window. Do *not* use the `text` function; instead, use the geometric shape primitives described on p. 43. Be inventive; use a variety of shapes and colors. (Note: `polyBezier` only works on Windows systems.)

5. Write a function

```
showArea :: IO
```

that (i) opens a 600x500 window; (ii) accepts 4 left-button clicks in the window describing a quadrilateral, and outlines this quadrilateral in blue; (iii) calculates the area of the quadrilateral; (iv) draws a red square of equivalent area centered at the origin; (v) waits for a space key to be pressed before closing the window. You can assume the user only enters quadrilaterals that are not self-crossing.

Import and use the `Shape` and `Draw` code from the Chs. 2 and 4 of the book (available on the course web site) to handle the area calculation and the generation of the result square. The graphics function

```
  getLBP :: Window -> IO Point
```

waits for a left button click and returns the corresponding mouse location in pixel coordinates. You'll need to write an inverse translation function from `Point` to `Vertex`.