Practical 1 — Programming with Simple Functions

The questions in this practical will be marked in the supervised laboratory session on Monday 13 October. Consequently, you should prepare for the laboratory sessions by attempting these problems in advance. You may work in groups of 1, 2 or 3 people. Half marks will be given for a correct solution. You should also write in comments why your solution is correct and half the marks will be given for your explanation and the beauty of your solution. We dont want essays but just to see you understand your code. Note marks will ONLY be given if all group members are present.

The two files that you will need for this exercise are available below. They are Quilt.hs and Prac1.hs. I suggest you make a new directory called Haskell, a subdirectory called Coursework1 and copy these files to this subdirectory.

This exercise is concerned with a novel datatype called *Quilt*. Elements of this type are patterned rectangles which can be reflected, rotated and joined together to create larger patterns. The basic functions on this type are

```
plain :: Int -> Int -> Quilt
fancy :: Char -> Char -> Int -> Quilt
flipV :: Quilt -> Quilt
flipH :: Quilt -> Quilt
rotate :: Quilt -> Quilt
sewV :: Quilt -> Quilt -> Quilt
sewH :: Quilt -> Quilt -> Quilt
width :: Quilt -> Int
height :: Quilt -> Int
border :: Char -> Quilt -> Quilt
```

The definition of these functions is given in the file `Quilt.hs`. The file `Prac1.hs` contains some auxiliary definitions. You will edit this file to contain your answers and, when you have finished, show them to the demonstrator at the lab. **Make sure you fill in the names and user id of the members of your group in the space provided in the file Prac1.hs**

Click on the file `ghci Prac1.hs` to start up GHCi - this will load the file `Prac1.hs`.

Evaluate the expression `eg 1` on the hugs command line to see an example of a quilt. Now try `eg 2` and `eg 3`. Use your favourite editor to view the Haskell files, and observe how these quilts were generated using the functions `plain` and

```
triangle 6

.-----
$.----
$$.---
$$$.--
$$$$.-
$$$$$.

pile 6

-----..-----
----.$$.----
---.$$$$.---
--.$$$$$$.--
-.$$$$$$$$.-
.$$$$$$$$$$.
```

Figure 1: Triangle and Pile

`fancy`. Try out some examples of your own to make sure that you understand what the basic functions listed above do. Ask if you need help on this. Also, what does the (infix) function `.` in `eg 5` do?

1 13 October Lab

1. Define a function `triangle :: Int -> Quilt` which for any n gives an $n \times n$ quilt with the pattern shown at the top of Figure 1.

2. Use the function `triangle` in the definition of a second function `pile :: Int -> Quilt` whose value on n is the $2n \times n$ pattern formed by sewing two triangles together; see Figure 1. The function `pile` uses a local declaration `tri`; look at `stripe` to see how this works—here, `q` is declared locally.

3. The pattern on the quilt in Figure 2 is given in two parts which are each formed by sewing piles onto a plain square.

   Define a quilt called `moneypiles` which looks like this.

4. Many interesting patterns can be obtained by rotation, e.g. by successively rotating a triangle, we get the windmill in Figure 3. Define a function `foursome :: Quilt -> Quilt` which given a square quilt of size n will produce the square quilt of size $2n$ formed by successively rotating the pattern through 90 degrees as above.

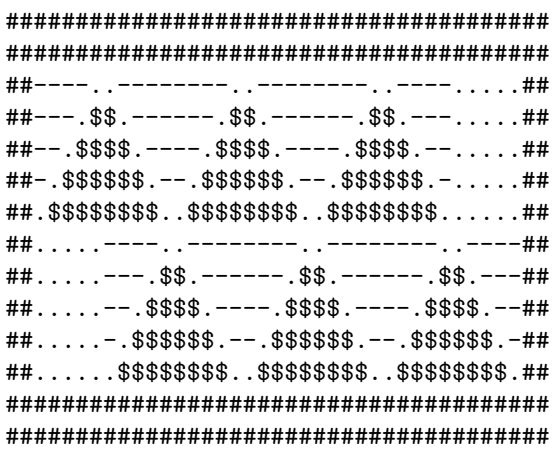

Figure 2: Moneypiles

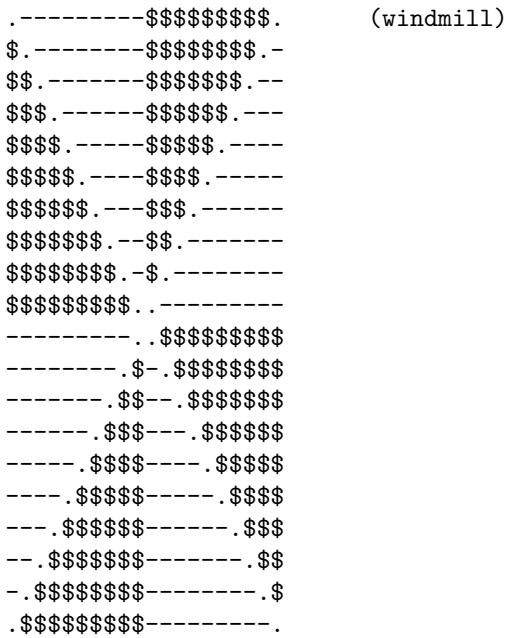

Figure 3: Windmill

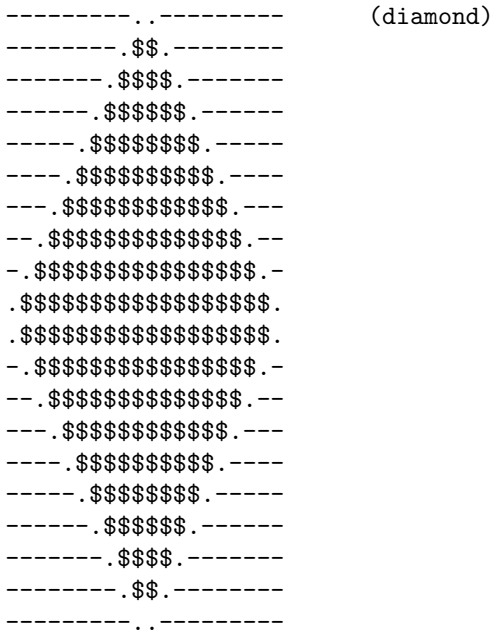

Figure 4: Diamond

5. Use this to give definitions of `windmill`, and also of the `diamond` in Figure 4.

6. Unfortunately, the function `foursome` will give an error when applied to a quilt that is not square because `sewH` and `sewV` are only defined for quilts whose edges match up. To overcome this, define a function `squareup :: Quilt -> Quilt` which turns any rectangular quilt into a square one by sewing on a plain patch of the appropriate size.

   Finally, to test things out, define the function

   ```
   flower n = foursome (squareup (stripe n))
   ```

   and you should find that this gives a rather pleasant flower pattern.