The aim of this tutorial is to help you familiarise yourselves with Haskell and Hugs. Try to write the types of all your functions, so as to get used to thinking within the confines of a strictly typed language. It is also best to write your programs as literate scripts, with ample comments in between functions. Remember that programming requires practice – just by working through these tutorial sheets as we go along, you will find that you need not do much more work for the final exam.

1. Define the function `circleArea`, which calculates the area of a circle with a given radius.

2. Define the function `dbl`, which given an integer as input, returns twice the value. Using `dbl`, define `quad`, which multiplies the input by four.

3. Define a function `sumN` which, given an integer `n`, recursively calculates the sum of the first `n` integers.

4. A rectangle can be described by giving the coordinates of the bottom left and top right hand corners.
   (a) Define a type synonym `Rectangle` to describe rectangles.
   (b) Define functions to calculate (i) the length of a rectangle; (ii) the breadth of a rectangle; (iii) the top-left, bottom-left, top-right and bottom-right points of a rectangle (as four separate functions); (iv) given a point and a rectangle, return whether the point lies inside the rectangle.
   (c) Define a function `areaR`, which, given a rectangle, returns its area.
   (d) Define a function `intersect`, which given two rectangles returns whether they intersect (hint: an easy way is to check whether a corner of one rectangle lies inside the other).

5. An trapezoid is a quadrilateral with two parallel sides. An orthogonal trapezoid is a trapezoid with two orthogonal internal angles:

   ![Trapezoid Diagram]

   (a) The area of an orthogonal trapezoid is \( \frac{1}{2} \delta (h_1 + h_2) \). Write a function which calculates this.
   (b) Haskell functions can be given other functions as parameters. We will be talking more about this later in the course. For the moment, try to define a function `approxArea1` which given a function (from `Double` to `Double`) calculates an approximation of the area under the graph between 0 and 1 by reducing it to a trapezoid:
The function definition would look something like:
\[
\text{approxArea1 } f = \ldots (f 0) \ldots (f 1) \ldots
\]
(c) Now define \texttt{approx1} which also takes the left and right bounds (which would replace 0 and 1).
(d) Define \texttt{approx2} which uses two trapezoids to approximate the area under a curve between two given bounds.
(e) Define \texttt{approxArea} which takes a number \( \varepsilon \), two bounds \( l \) and \( r \), and a function \( f \), and then, if \( l \) and \( r \) are sufficiently close to each other (their difference is less than \( \varepsilon \)) returns the area using \texttt{approx1}, otherwise splits the range into two (from \( l \) to \((1+r)/2\) and from \((1+r)/2\) to \( r \)) and returns the sum of the approximate areas for both ranges (using \texttt{approxArea}) itself.