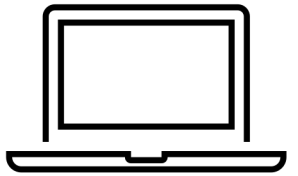
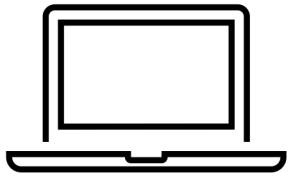
Give it a go!



- Download basics.hs from Blackboard and load it into the Haskell interpreter.
- *Try* those things that were presented in the first lecture:
  - :browse
  - add 2 3
  - double (add 2 3)
  - double add 2 3
  - inc 1
  - inc (inc 1)
  - inc2 1
  - inc2 (inc2 1)
  - inc (inc2 1)
- Which ones work? Which ones don’t? Why don’t they work?

Give it a go!



1. What are the types of the following functions?

```
second xs = head (tail xs)
swap (x,y) = (y,x)
pair x y = (x,y)
```

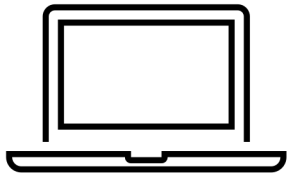
```
double x = x*2
palin xs = reverse xs == xs
twice f x = f (f x)
```

Challenge: What do they **do**? What do the library functions used in them do?

2. Implement a function for computing the Euclidean distance between two points, (x1,y1) and (x2,y2).
   Reminder: the Euclidean distance is given by:

$$\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Give it a go!



3. Define a function that uses library functions to return the first word in a string. For example:

```
> firstWord "a test string"
"a"
>firstWord "   the trickier test string"
“the”
```

Remember to include a type for your function.

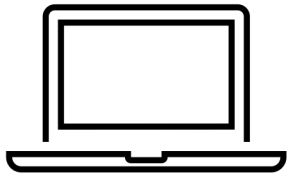
4. Using library functions, define a function

```
halve :: [a] -> ([a],[a])
```

that splits an even-lengthed list into two halves. For example:

```
> halve [1, 2, 3, 4, 5, 6]
([1, 2, 3], [4, 5, 6])
```

Give it a go!



5. Write a function that takes four arguments, which are meant to be the grades for four equally-weighted modules. The function should calculate the average grade for these inputs, then output the appropriate text:
    - “H1” for an average of 70+
    - “H2.1” for an average in the range 60 – 69
    - “H2.2” for an average in the range 50 – 59
    - “H3” for an average in the range 45 – 49
    - “Pass” for an average in the range 40 – 44
    - “Fail” for an average < 40

    Try to produce two solutions: one using conditional expressions, the other using guarded expressions.

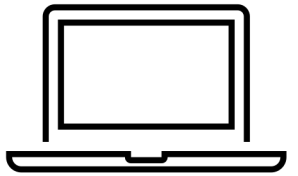
Lists

- There are many library functions and several operators that work on lists. A couple of useful list operators:
  - !! (`xs!!n` returns the n^{th} element of xs, starting at 0)
  - ++ (`xs++ys` returns a single list of all the elements of xs followed by all the elements of ys)
- It is easy to *generate* lists in Haskell:

```
ghci> [1..10]
[1,2,3,4,5,6,7,8,9,10]
```

This generates at list starting at 1, incrementing by 1 for each value, stopping as soon as it reaches a value >= 10.

Give it a go!



8. You wrote a function earlier to calculate the classification of the average grade over four marks. Create an improved version that can calculate the classification of the average grade over a list of marks. Assume the modules are equally-weighted.

9. Challenge: instead of a list of marks, your input should be a list of (mark, credits) pairs, where the second item in the tuple indicates how many credits the mark is associated. The output should be the classification of the weighted average grade over this list.