# Worksheet 4 Function application

**Task 1**

1. The process of giving particular inputs to a function is known as:

2. Without using a computer, think about the function

 times x y z = x \* y \* z

 If we then define

fiveDeep l w = times 5 l w

 What is returned by the following?

 fiveDeep 3 4

3. Create and load a file with the functions **times** and **fiveDeep.** What happens when you enter **:type** for each of these at the prompt? Why?

### Task 2: map and filter

4. Write down what you expect to the returned by the following. Use a calculator if you need to. Then test in Haskell.

 (a) map (\*5) [4,9,15]

 (b) map sqrt [4,9,16,37]

 (c) map sin [0,pi/4, pi/2]

5. Create and save a file filter.hs to implement the function

 isEven n = n `mod` 2 == 0

 (Hint: Use the backward quotes on the left of the 1 key on the keyboard to surround the mod operator. ) Write your function below.

 Use **map isEven** to test it on a list of half a dozen integers. What is the result?

 What would you expect to happen if you tried an argument which is not an integer? Try it.

6. Explore the use of **filter (>"g")** with a list of characters, e.g. ["A","B","a","b","h","n"]

What is output by the function?

### Task 3

7. Use a fold operation to find the sum of the elements in [1,2,3,4,5]

8. Adapt your code from question 6 to find the factorial of 5, i.e. (5 x 4 x 3 x 2 x 1)

A useful data type in Haskell is an ordered pair, called a **tuple**. Some examples are:

(1,2), (5.8, True) and (“Helen”, 15)

9. Define a tuple **t1** from any of these examples. Try out the built-in functions **fst** and **snd** on t1. The functions return the first and second elements respectively

10. Define your own list of tuples, with a consistent type or pair of types. Write down the list, and the result of applying **map fst** or **map snd** to your list.

11. Define a list of tuples of pairs of numbers. Use a **fold** function to calculate the sums of the first elements from every tuple.

12 Write functional programming code in Haskell or another language, including type declarations, which takes a list of the first 100 integers and returns the odd numbers which are multiples of 3.

 Hint: the definition **listN = [1..N]** defines a list of all the integers between 1 and N.