The University of Melbourne
School of Computing and Information Systems
COMP30026 Models of Computation

Sample Answers to Tutorial Exercises, Week 1

3. We need to move from the lower left corner to the upper right. It does not really matter whether we are moving between grid points or we move from square to square—the two variants are isomorphic. Here is a recursive solution:

```
paths_rec :: Integer -> Integer -> Integer
paths_rec 1 m = 1
paths_rec n 1 = 1
paths_rec n m = (paths_rec (n-1) m) + (paths_rec n (m-1))
```

   If you are not used to recursion as a problem solving technique, try to ponder this definition and make sure you understand it. It says that, if there is only one row or only one column, then there is only one path. Otherwise we have a choice for the first step we make, namely move up or to the right. Do you see how the third equation therefore calculates the number of possible paths in that case?

   We wrote the solution in Haskell, so that we can actually *run* it, but we could also just have written it as recursive definition of a mathematical function, if you prefer that:

$$p(n,m) = \begin{cases} 1 & \text{if } n = 1 \text{ or } m = 1 \\ p(n-1,m) + p(n,m-1) & \text{otherwise} \end{cases}$$

   It is a nice and elegant solution, but unfortunately, as a program, it runs in exponential time. So perhaps we should analyse the problem again. Maybe we can find a *closed form* solution, by which we mean one that does not use recursion? Thinking . . .

   The paths that we are enumerating all have $(m-1)+(n-1)$ steps, of which $m-1$ are horizontal steps and $n-1$ are vertical. So it is really a matter of deciding which of the $m+n-2$ steps are the horizontal ones. Think of the steps being numbered from 1 to $m+n-2$; we are asking how many ways there are to choose $m-1$ of those numbers. Well, there are K^{m+n-2}_{m-1} ways. We can easily write a function `k` for "n choose m", that is, K^n_m:

```
paths :: Integer -> Integer -> Integer
paths n m = k (n+m-2) (m-1)

k n m = numerator `div` denominator
  where
    numerator   = product [(n-m+1)..n]
    denominator = product [1..m]
```

4. (a) Here is a solution: $\begin{array}{|c|c|c|}\hline \frac{\texttt{aaa}}{\texttt{aa}} & \frac{\texttt{baa}}{\texttt{abaaa}} & \frac{\texttt{aaa}}{\texttt{aa}} \\ \hline \end{array}$
   (b) No solution is possible. Why?
   (c) No solution is possible. Why?

5. Okay, this is a trick question, because there is no such algorithm! Towards the end of the course, we return to this phenomenon ("undecidability"). It pops up not only in puzzle problems such as Question 4's, but also in many practical problems, such as most of the problems faced by designers of optimizing compilers.