Assessed Exercise – Part II (General)
Addendum: Clarification and Hint
Introduction to Functional Programming

A Dawar

Issued: 4 February 2000

This document is intended to serve as an addendum to the assessed exercise that was handed out on 1 February. It offers correction, clarification and a hint on solving part 3 of the exercise.

In part 3 of the assessed exercise, you were asked to define a function that computes a knight's tour from any starting location.

1. It is stated there that a knight can begin at any position on the board and in exactly 64 moves return to that starting location, having visited every square on the board. While this is true, searching for such a tour may be computationally too expensive. For the purpose of the assessed exercise, it will suffice if you find a tour in which the knight visits every square on the board exactly once *without necessarily being able to return to the original square*. So, the tour would involve 63 moves, visiting all 64 squares (including the starting position).

2. A simple minded search strategy (without some guiding heurisitics) may be unlikely to find a tour in a reasonable amount of time. If you produce a program which carries out such a search and would, in principle, find a tour, but you are unable to run it for long enough to actually find one, you will receive some (but not complete) credit. For this, however, if you are unable to produce output showing that your program works, you must give a clear *convincing* explanation of why your program would work.

3. One search heuristic that is known to find solutions for the knight's tour problem quickly is the following. From any given position, to choose among the possible next positions, first try the one which has the fewest positions that it can be extended to. For example, if the knight is currently at position a, and the valid next positions that it can proceed to from a are b, c and d, and moreover, from b there are 3 valid positions that can be reached, from c there are 2, and from d there are 6, then you should first try c.