1 Foundations of Computer Science (LCP)

A one-person game (such as Rubik's cube, or peg solitaire) has a finite number of possible *states*, some of which count as *winning*. A *move* is a step from one state to another. From each given state, the player can choose from a set of (zero or more) possible next moves. We call a state *winnable* if a winning state can be reached from it in zero or more moves.

For simplicity, assume that states are coded as integers. Also assume that we are given functions `winning(x)` returning true or false and `next(x)` returning the list of states that can be reached in one move from state `x`.

(*a*) The following code is an attempt to implement the notion of winnable:

```
fun exists p []      = false
  | exists p (x::xs) = p x orelse exists p xs;

fun winnable x = winning x orelse exists winnable (next x);
```

Briefly explain how this code works. Also describe its main limitation: how it can fail to find a winning state that is only a few moves away. Illustrate this point by giving specific definitions of `winning` and `next`. [5 marks]

(*b*) Modify the code above to yield the function `winpath x`, which returns the list of states from `x` to the winning state found or, alternatively, the empty list to indicate that no winning state was found. [4 marks]

(*c*) Sometimes we are only interested in a winnable state if it is only a few moves away from the current state. Modify your solution from part (*b*) to obtain the function `bounded_winpath n x`, which looks for winning states that are at most `n` moves away from `x`. [3 marks]

(*d*) Use your solution from part (*c*) to obtain the function `new_winpath x`, which has the same objective as `winpath x`, but without the limitation mentioned in part (*a*). Briefly explain why the limitation no longer applies and the price that has been paid for this. [5 marks]

(*e*) Briefly outline an alternative approach to correcting the limitation mentioned in part (*a*), using the notion of a queue. What are the advantages and drawbacks of this approach? [3 marks]

For full credit, code should be concise and clear. Exceptions may be useful in this but are not required.