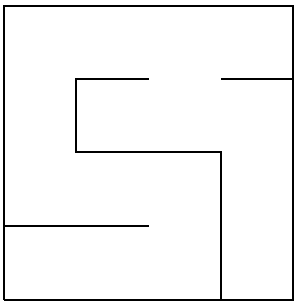
Question 1

A rectangular maze is represented by a two dimensional array (list of lists) of strings. The strings have any length from 0 to 4 characters which may be "n", "s", "w", or "e", and tell you which directions it is possible to move in from each position.

For example, this trivial maze



is represented by this array

```
[ [ "se", "we", "wse", "w"  ],
  [ "ns", "e",  "nwe", "sw" ],
  [ "ne", "ew", "ws",  "sn" ],
  [ "e",  "ew", "wn",  "n"  ] ]
```

The rows are numbered from 0 at the top increasing downward and columns from 0 at the left increasing to the right

The question is, given the row and column of the starting position `(rs, cs)` and the ending position `(re, ce)`, what moves do we have to make in order to get from start to end as quickly as possible (i.e. smallest number of moves)?

Design a program that could print a minimum-length sequence of moves as a string made from the letters "n", "s", "w", and "e", one per move. In the example, start `(1, 0)` and end `(3, 3)` the solution would be "neesess".

You must use an heuristic search in which the heuristic function is the straight line (Pythagoras) distance from the current position to the end position.

You do not have to write the program, that might take a while. You do not even have to do it in "pseudocode", whatever that really is. But you must say exactly how it will work: the data structures needed, and a detailed outline of the algorithm. It should be possible for a programmer with no A.I. knowledge to follow your solution and produce a working program.

Question 2

Your task here is to design an algorithm that can solve the farmer+fox+chicken+corn problem using a breadth-first search.

As a reminder: only the farmer can row the boat; the farmer can take at most one thing with him in the boat; the fox and the chicken can not be left together without the farmer; the chicken and corn can not be left together without the farmer.

You must find a solution using breadth-first search.

You do not have to write the program, that would take too long. You do not even have to do it in “pseudocode”, whatever that really is. But you must say exactly how it will work: the data structures needed, and a detailed outline of the algorithm. It should be possible for a programmer with no A.I. knowledge to follow your solution and produce a working program.

Question 3

Give an efficient python implementation of a priority queue. It must support these operations:

`create()`	create an empty priority queue ready for use.
`empty()`	→ `True` or `False`
`add(d, p)`	add data item `d` with priority `p`
`take()`	→ `(d, p)` the data item and priority, `p` being the smallest still in the priority queue

The ability to change the priority of an item already in the queue is not needed.

If you give a full implementation of `add`, you only need to give a complete and accurate description of how `take` would be implemented. Conversely, if you give a full implementation of `take`, you only need to give a complete and accurate description of how `add` would be implemented. You must however give a full implementation of `create`, `empty`, and at least one of `add` or `take`.

You may use any kind of underlying basic data structure you like: a linked list, a python [ ] list, whatever. But nothing that does a lot of the work for you, you definitely should not need an `import` statement.

Efficiency is required. For example, searching for the end of a linked list is not efficient. On lists, slices `[a:b]`, `+`, `+=`, `.pop(n)`, `.remove(item)`, etc., are not efficient. An inefficient action that is only used very rarely is acceptable, but it is not allowed every time any particular operation is performed.

Question 4

a. Have you checked your answers?

b. What is 6×9?