Workshop exercises set 2.

QUESTION 1
Give a high level description (not programming language specific)
of at least five different possible representations of playing cards
from a standard 52 card deck. Describe the advantages and disadvantages
of each representation.

The standard 52 card deck has 13 cards in each of four suits.
The suits are clubs, diamonds, hearts and spades, and the 13 ranks in
each suit are the 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen, king and ace.
In this question, we ignore jokers.

QUESTION 2
Define a Haskell type for representing "font" tags in HTML. A font tag
can specify zero or more of the following: the size in points (e.g. 10),
the face (e.g. "courier") and the colour. The colour can be described
using a colour name (e.g., "red"), a six−digit hexidecimal number
(e.g. #02EA1F) or a RGB triple of numbers (e.g. rgb(255,100,0)).

Note: the font tag is the most widely misused of all HTML tags,
and in fact it is fundamentally misconceived. The font should be up to
the VIEWER of the web page, not the web page DESIGNER; if the designer
selects a small font, people with bad eyesight looking at the page
won't be able to read it. This is why the font tag is actually deprecated,
which means it is slated to disappear in a future version of the HTML standard.

QUESTION 3
Implement a function 'factorial' that computes the factorial of a given
integer. Include a type declaration.

QUESTION 4
Implement a function 'myElem' which returns True if a given item is present
in a given list. Include a type declaration.

QUESTION 5
Implement a function 'longestPrefix' which returns the longest common prefix
of two lists. ie: When applied to "extras" and "extreme", the function
should return "extr".

QUESTION 6
Without necessarily understanding the code, translate the following
C function into Haskell.

```
int mccarthy_91(int n)
{
    int c = 1;

    while (c != 0) {
        if (n > 100) {
            n = n − 10;
            c−−;
        } else {
            n = n + 11;
            c++;
        }
    }

    return n;
}
```

QUESTION 7
Write a Haskell function which takes two integers, min and max, and
returns a list of integers from min to max, inclusive. Note there are
two different strategies to solve this problem: we can build up the list
from min to max or backwards, from max to min. How does your Haskell
code compare with a version in an imperative language such as C, and how
would you reason about the correctness of a C version?