Functional Programming coursework

This coursework has three parts.

(a) Provide an algebraic data type definition to represent the four suits in a deck of cards. Make use of this algebraic type in a type synonym to represent a particular card, for example "the ten of hearts".

(b) Write a function that will take as arguments (i) a number and (ii) a list of elements. Each element in the list represents a card (using the types defined in (a)). The function should produce as its output a "shuffled" version of the input list. The function should shuffle the list of cards as many times as is indicated by the first argument.

The action of shuffling should cut the deck in half and then interleave the cards, with the previous top card now being the second card in the pack. For example, if a list of four items A, B, C and D is shuffled once the result should be C, A, D, B. If that result is shuffled again the result should be D, C, B, A. As a simplification, you may assume that where there are an odd number of items in the list, the last item should be discarded.

Your function should detect the case where there are more than 52 elements in the input list, which it should treat as an error.

(c) Write a function which takes two integer numbers and generates a result using the following rules:

1. Multiply the two numbers together
2. Then add all the digits of the result
3. If the sum of the digits has itself only one digit then return it as the result of the function, otherwise repeat from (2).

For example, if your function is called "**f**", a sample interaction with the Miranda system would be:

Miranda f 3 4
3
Miranda f 7 7
4
Miranda f 30 19
3

The last of the above examples is calculated as follows:
*30 * 19 is 570;*
5 + 7 + 0 is 12 (which has 2 digits);
1 + 2 is 3 (which has I digit);
the result is 3.

Submission deadline and procedures are not yet finalized and will be announced very soon - please assume that you have about 10 days to do this work.

The coursework will be binary marked - any reasonable attempt will get an A, and anything else will fail. The concept behind binary marking is that the coursework is "formative" rather than "summative" - i.e. it is more important that you actually engage in some practical programming, and therefore you have some practical skill, and the coursework is not attempting to distinguish different levels of skill.