2004 Paper 6 Question 10

Foundations of Functional Programming

Continuation passing allows lambda calculus and functional languages to describe many forms of sequential control structure. If you write code in your answer to this question you may write it either in lambda-calculus or in an equivalent ML-like syntax, but in the latter case you must not rely on ML's order of evaluation.

(*a*) Using the continuation passing style, show how to model the following impure code without use of `ref` or `:=` but so that the exact sequence of values that `k` takes during a calculation still arise.

```
val k = ref 0;
fun c(n, r) = (
    k := !k + r;
    if r=0 orelse r=n then 1
    else c(n-1,r-1) + c(n-1,r));
```

You may suppose that conditions and basic arithmetic are available to you as existing primitives, so `if`, `orelse` and `+` can all appear in your answer, even though `!`, `:=` and the use of `;` that indicates sequential execution must not. [14 marks]

(*b*) Discuss how exception-handling mechanisms such as ML's `raise` and `handle` can be mapped onto uses of continuations. Construct a small example to illustrate your explanation. [6 marks]

All code you write will be expected to be annotated so that it is easy for a human reader to see what it sets out to achieve: clarity of exposition will be considered much more important than exact syntactic validity in any particular programming language's syntax.