SOLUTION NOTES

As submitted to the Examiners in January 2001.

Foundations of Functional Programming 2001 Paper 6 Question 10 (ACN)

```
Students who express opinions coherently get the marks even if their
view is not mine as expressed here.

(1) Polymorphic
    Ability to have types parameterised by (universally
    quantified) type variables. Useful for eg lists and arrays
    where items in list must all be same type but who cares what it is
    when we just handle the list itself.
    Want equivalent in ordinary languages. C almost has void *, Java
    almost has Object but neither of those quite gets there! Object
    hierarchies are a sort of alternative!
(2) Type reconstruction
    User does not have to specify types - compiler can re-create them.
    May save lots of dull user input while keeping code type-safe. Good!
    functional languages often very rigid and disciplined and this helps
    make it possible. Best if types are jolly disjoint, so probably tough
    in an O-O world! But would be nice to have as much as poss! Actually
    it is tough technology.
(3) Higher order
    Functions as values, results etc. Use with map/filter etc and in
    curried fns for partial evaluation. Major style feature in functional
    prog. MAYBE ordinary languages would gain a lot by better provision
    here [but ACN knows about spaghetti stacks and the pain of full
    implementation!]
(4) Lazy
    evaluate only on demand. streams, lazy lists => "infinite" data
    structures. Are these serious or just a nice joke?
    IN ordinary language the imperative bit and delayed eval could
    fight very seriously!
(5) Continuations
    fn that "does not return" exits by invoking its continuation. Use to
    model many elaborate control structures.
    Maybe mainline languages have those control structures already so that
    the sane uses of continuations are already catered for!
```