SOLUTION NOTES

Foundations of Computer Science 2002 Paper 1 Question 6 (LCP)

Parts (a) and (b) refer to Lecture 3, O notation. For (a) candidates need not repeat the formal definition, but they should know that a function is in $O(g(n))$ provided the cost of interest is bounded by $|g(n)|$ for all but perhaps finitely many values of n, and where g can if necessary be scaled by some constant. Naturally n is some measure of the input size, and some specific cost (such as time, space or comparisons) is being measured by O-notation. They should probably note that the constant factor makes the measure independent of transient factors such as hardware speed, while ignoring finitely many values of n focusses attention on the long-term efficiency trends.

(b) Here is the ranking:

$$O(\log n) \qquad O(n^{1/3}) \qquad O(5n^2) \qquad O(n^3 - 3n^2) \qquad O(n2^n) \qquad O(e^n)$$

Candidates could note that $O(5n^2)$ is $O(n^2)$ and that $O(n^3 - 3n^2)$ is $O(n^3)$, so the ranking simplifies to

$$O(\log n) \qquad O(n^{1/3}) \qquad O(n^2) \qquad O(n^3) \qquad O(n2^n) \qquad O(e^n)$$

Most of the comparisons require little justification. For example, each doubling of n increases $\log n$ by one while increasing $n^{1/3}$ by a factor of $2^{1/3}$, so clearly the latter will permanently overtake the former after a while. Presumably most candidates can remember that $e > 2$.

(c) Here is one solution:

```
fun bsearch f (y: int) =
  let fun search (a,b) =
      if a=b then b
      else
        let val c = (a+b) div 2
            val y' = f(c)
        in  if y' = y then c
            else if y'>y then search (a,c)
            else (* y'<y *)   search (c+1,b)
        end
  in search end;
```