12 Optimising Compilers (AM)

The following C-like program reads a `List` of integers by making calls to external procedures `readint()` to read and return an integer and `cons()` to construct a `List`:

```
    struct List { int hd; List *tl; };
    List *readlist() {
L0:   List *p = 0, *q, *t;  int v;
L1:   // comment for Part (e)
L2:   v = readint();
L3:   if (v < 0) return p;
L4:   t = cons(v, 0);
L5:   if (p == 0) {
L6:       p = t; q = t;
      } else {
L7:       q->tl = t; q = t;
      }
L8:   goto L2;
    }
```

(*a*) Define the concept of a variable being live at a program point, distinguishing a semantic notion from one which is more practically computable. Sketch an algorithm to compute the set of variables live at each program point. [5 marks]

(*b*) Give the sets of live variables your algorithm would compute for the program points `L0` ... `L8`. Also indicate any discrepancies between these and sets derived from your semantic notion of liveness. [5 marks]

(*c*) Does the above code have any data-flow anomalies? If so explain what they are and what action a helpful compiler might take. [2 marks]

(*d*) Suppose the above code were compiled for an ARM-like processor which assumes registers 0–3 are corrupted by procedure call and registers 4–7 are preserved by procedure call; register 0 is also used for the first argument to and result from a procedure call. Assuming a register allocator operating by graph colouring, which registers might be allocated to variables `p`, `q`, `t` and `v`? Indicate when there is a choice between equivalent allocations and when there is a benefit of using one register over another. What, if anything, would be wrong with simply allocating `p`, `q`, `t` and `v` respectively to `r4`, `r5`, `r6` and `r7`? [5 marks]

(*e*) Now suppose that the commented line `L1` were replaced with some more complex code (which does not use `p`, `q`, `t` or `v`) and the register allocator found that 9 registers were now required for `readlist()` when only `r0` ... `r7` are available. How would an allocator proceed for the code as written? Can you suggest an adjustment to the source code (respecting your company's coding requirement that all declarations, but not necessarily initialisation, occur at the very start of a procedure) to allow all variables to be allocated a register? [3 marks]