11 Optimising Compilers (tmj32)

The following C-style code from an untyped language is analysed by a compiler, where the `work()` function is assumed to have no side effects.

```c
1    c = &b;
2    *c = &c;
3    a = c;
4    c = &d;
5    if (v == 0)
6        *c = **a;
7    else
8        *c = *b;
9    *a = &a;
10   work(a);
11   work(c);
```

(a) Describe alias analysis and the transformations it enables. [4 marks]  

(b) Summarise Andersen’s analysis and calculate the points-to set, $pt(x)$, for each pointer, $x$, within the C-style code above. [9 marks]  

(c) Describe the reason that the analysis overestimates some of the sets in the answer to Part (b). [2 marks]  

(d) Now assume that the `work()` function may alter memory locations reachable through its argument. Explain why the two calls to `work()` in lines 10 and 11 cannot be executed concurrently using the analysis from Part (b), but can be based on the answer to Part (c). [5 marks]