Compiler Construction 2005 – Paper 6 Question 5 (AM) Solution Notes

This question is about section 9 and 10 of the notes.

(*a*)

```
int a = 1;
int f() { return a;}
int g(int a) { return f(); }
print g(2);
```

(*b*) Static chains within the stack and heap-allocated closures containing values of free variables. The former has stack lifetime of storage for the static chain, and hence requires language restriction to forbid returning of function values beyond the scope of any of their free variables. Latter uses heap allocation (unlimited lifetime) and so function values may be so returned, e.g. ML

```
int f x y = x+y;
int g = f 1;
int h = f 2;
print g 10 + h 11;
```

In my definition, free variables are *naturally* associated with closures by Lvalue using static chains (location shared) but by Rvalue when using free-variable lists, because the values of free variables are copied into the closures. (However, the alternatives may be implemented by a little more work – but this isn't asked for.)

(*c*)

```
int f(g,n) = n==0 ? 1 : n * g(g, n-1);
print f(f,7);
```

(*d*)

```
int a = 0, b;
void f(x) { x++; b=a; x--; }
void g(x) { x+=2; }
f(a); // b=0 -> CBV/CBValRes; b=1 -> CBRef
g(a); // a=0 -> CBV; a=2 -> CBValRes/CBRef
print a+b;
```

(*e*)

```
(p) offset 0: a   // field 1
    offset 4: b   // field 2
    offset 8: f   // virtual function f stored as pointer
(q) offset 0: a   // field 1 -- nec. for inheritance
    offset 4: b   // field 2 -- nec. for inheritance
    offset 8: f   // virtual function f -- nec. for inheritance
    offset 12: c  // field 3
    offset 16: g   // virtual function g
```