1999

p4q4
AM

Compiler Construction p11q4

Compiling Techniques cmptech2.tex

(a) It is commonly suggested that Algol-60 call-by-name can be modelled by passing a function as a call-by-value parameter. Show how a program containing a definition

```
int f(int x:name) { ... x ... x ... }
```

and a call f(*e*) to f can be replaced by an equivalent definition and call using only call-by-value. [6 marks]

(b) Most such explanations assume that the uses of x within f only occur in Rvalue context. However Algol-60 also permits the equivalent of

```
int g(int x:name) { if (p) { ... x := x+1; x := -x; ... }
                    return x;
                  }
```

and calls like g(a[k()]) which, when p is true, would have the effect of calling k() five times and consequent access to five (possibly different) subscripts of array a[]. Develop your explanation for part (a) to cover also the case of a call-by-name parameter being used in both Lvalue and Rvalue contexts. [Hint: note that when p is false then the actual parameter to g need not be an Lvalue, so you may need two 'thunks'.] [8 marks]

(c) Using part (b) or overwise, give a translation ~~of~~ of a definition and call h(*e*) using call-by-value-result (Ada in out mode) with no uses of the address-of (&) operator other than those involved in call-by-name. Your explanation is allowed to deviate from call-by-value-result by allowing side-effects in *e* to take place twice. [6 marks]

Model Answer

(a) and (b)

```
int f(int x:name) { ... x := x+1 ... }
... f(e)...
==>
int f(int xr(): value, int *xl(): value) { *xl() := xr()+1; }
... f(fn () => e, fn ()=> &e) ...      if e is an Lvalue
... f(fn () => e, fn ()=> NULL) ...    if e is not an Lvalue
```

(c)

```
int f(int x:value result) { ... }
... f(e)...
==>
int f(int xx: name) { int x=xx; { ... } xx = x; }
... f(e)...
```