

Input: Variable *var* at code in function *func*

Output: A list of post-deallocation *macros* suggested for *code* in *func*

```
1: Variables
2:   MutateAnalyser: Read/Write Analyser
3:   ReturnAnalyser: Return analyser
4:   LiveAnalyser: Live variable analyser
5:   CopyAnalyser: Copy elimination analyser
6:   macros: a list of macros suggested for code by the deallocation analyser
7: end Variables
8: procedure CHOOSEPOSTDEALLOCMACRO(code, func)
9:   macros := [] // Store all macros used in code
10:  if code is Assignment then
11:    lhs = left-handed side of code
12:    rhs = right-handed side of code
13:    // Check if copy of rhs is removed
14:    isCopyRemoved  $\leftarrow$  CopyAnalyser.isCopyRemoved(rhs, code, func)
15:    if isCopyRemoved then
16:      macros.append(TRANSFER_DEALLOC(lhs, rhs))
17:    else
18:      macros.append(ADD_DEALLOC(lhs, rhs))
19:    end if
20:  else if code is Function call then
21:    lhs = target variable of called function
22:    callee = called function
23:    for each param in code do
24:      if param is Array then // Macro is applied array type only
25:        isLive  $\leftarrow$  LiveAnalyser.isLive(param, func)
26:        isMutated  $\leftarrow$  MutateAnalyser.isMutated(param, callee)
27:        isReturned  $\leftarrow$  ReturnAnalyser.isReturn(param, callee)
28:        // Choose macro based on above analysis results
29:        switch isMutated – isReturned – isLive do
30:          case  $F - F - T \vee F - F - F \vee T - F - F$ 
31:            macros.append(RETAIN_DEALLOC(lhs, param))
32:          case  $F - T - F \vee T - T - F$ 
33:            macros.append(RESET_DEALLOC(lhs, param))
34:          case  $F - T - T \vee T - T - T$ 
35:            macros.append(CALLER_DEALLOC(lhs, param))
36:          case  $T - F - T$ 
37:            macros.append(CALLEE_DEALLOC(lhs, param))
38:          end if
39:        end for
40:      else if code is Return then
41:        ret = return variable of code
42:        for each var variable in func, except for ret variable do
43:          if var is Array then
44:            macros.append(PRE_DEALLOC, var)
45:          end if
46:        end for
47:      else
48:        // No needs to use Macro
49:      end if
50:    end procedure
```