Programming Language Comprehensive, Feb 1995
Answers

Question 1:

```
{n>=m}
x:=0;
i := m-1;
{x = number of zeroes in interval b[m]...b[i]}  Invariant
while i <> n do
begin
  i:=i+1;
  if b[i] = 0 then x:=x+1;
end
{x = number of zeroes in interval b[m]...b[n]}
```

Question 2: The state refers to the contents of memory, or more specifically to the (current) contents of storage variables referred to by a program. The environment is an association of meanings to identifiers, or in syntactic terms often represented by a symbol table.

The difference is that assignment statements change the state, whereas definitions, such as procedure declarations, change the environment. Furthermore, the change effected by assignment is (generally) irreversible, is persistent, and destructively alters variable contents. In contrast, a local declaration creates a "new" binding for an identifier, but does not destroy a previous binding; the previous value is saved, and sometimes reused later.

Question 3: (A) c_0 is everywhere-undefined.

$$c_{i+1}s = \begin{cases} s & \text{if } s(n) \leq 0 \\ c_i(s \mid k \mapsto k * x, n \mapsto n-1) & \text{otherwise} \end{cases}$$

(B). The graph of c_i (it's set of input-output pairs) is a subset of the graph of c.

(C). The graph of c is the union of the graph of all the c_i's.

Question 4 (A) In call-by-name, the effect of a function call is the same as the effect that would be obtained by replacing all formal parameters with the actual parameters. So, $(\lambda x, P)Q = P[Q/x]$.

In call-by-value, the arguments to a function call are evaluated, to produce values, before being bound to formals. So, given $(\lambda x.P)Q$ we first reduce Q, obtaining a value V, if Q terminates, and then the call behaves like $P[V/x]$. In the presence of side-effects, we must also take into account evaluation order if there are several parameters, and there must be a state parameter to describe evaluation.

Call-by-need is like call-by-name in that it does not evaluate its argument first. However, once the argument is used once, its value of saved and reused on further accesses to the corresponding formal parameter. Thus, needless repeated evaluation is avoided.

(B) ((lambda (x) 1) (T))
If T is a term that infinite loops, this expression will infinite loop under CBV but terminate under CBName. We can code T easily using recursion, but not assignment.

(C) ((lambda (x) (+ x x)) (begin (print 5) 2))
Under CBName this will print 5 twice and return 4. Under CBNeed it will print 5 once and return 4.

Question 5.

Stack allocation is used for local definitions, such as procedure declarations. It is also used for certain locally-declared variables, that are de-allocated on exit from a block.

The heap is used for pointers, which are dynamically created, and whose lifetime (extent) can exceed the scope in which they are first created.

Question 6. See Aho-Sethi-Ullman, pp181, 182.