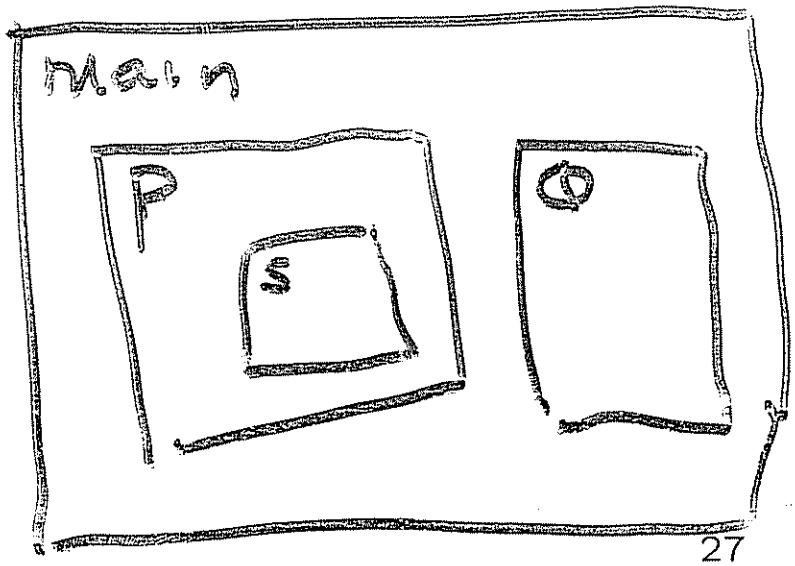
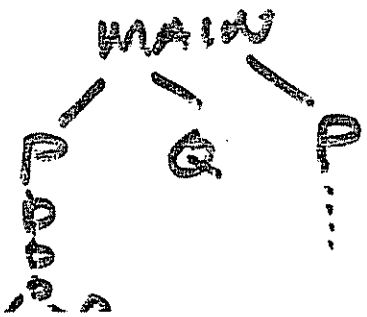
Example

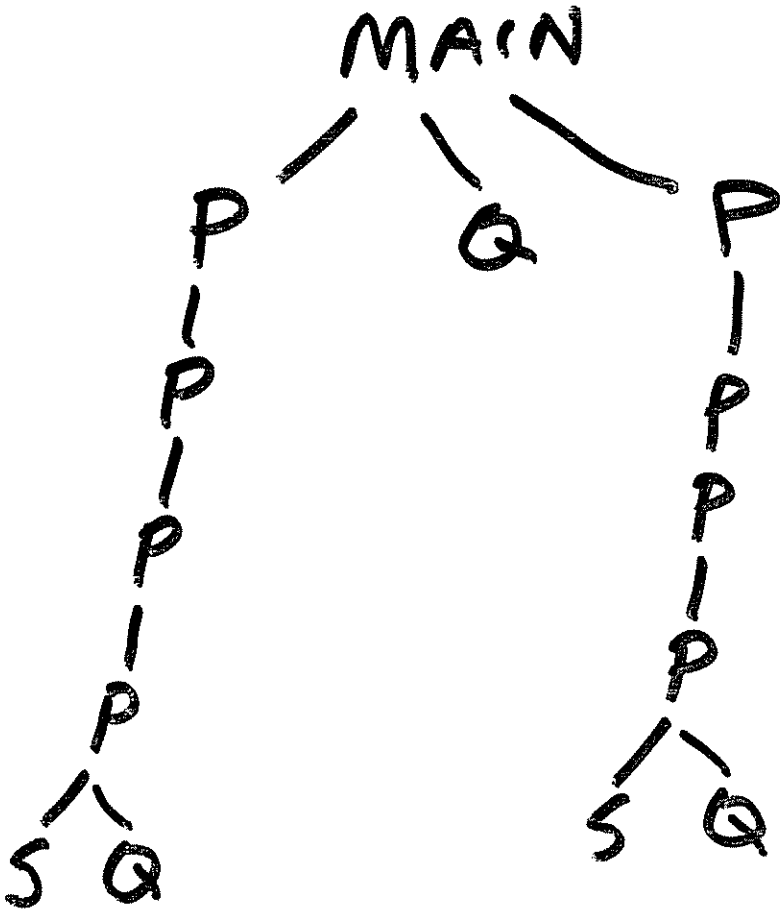
```
main
  procedure P
  begin
    procedure S begin ... end S;
    if random(1) < 1 then P()
    else { S(); Q() }
  end P;
  procedure Q begin ... end Q;
  P;
  Q;
  P;
end
```

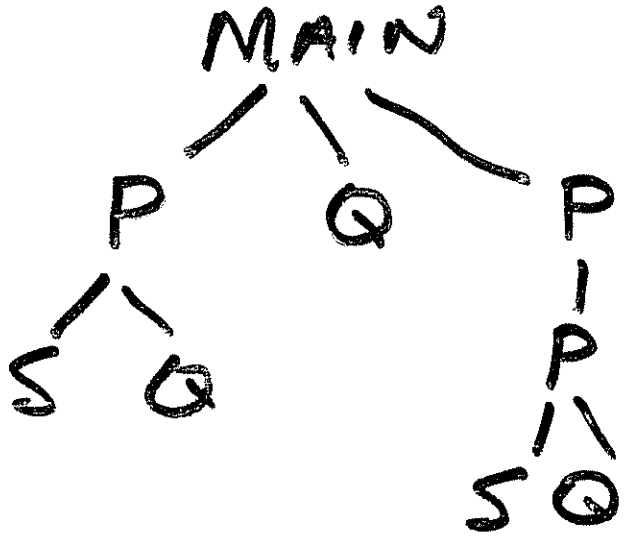
Note Recursive

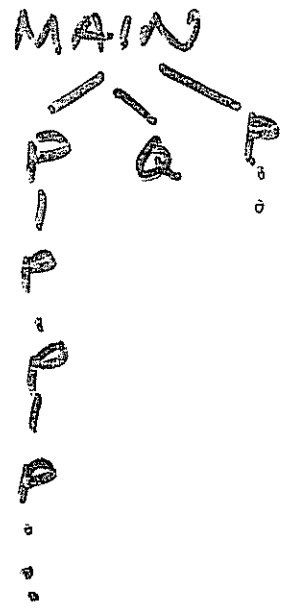


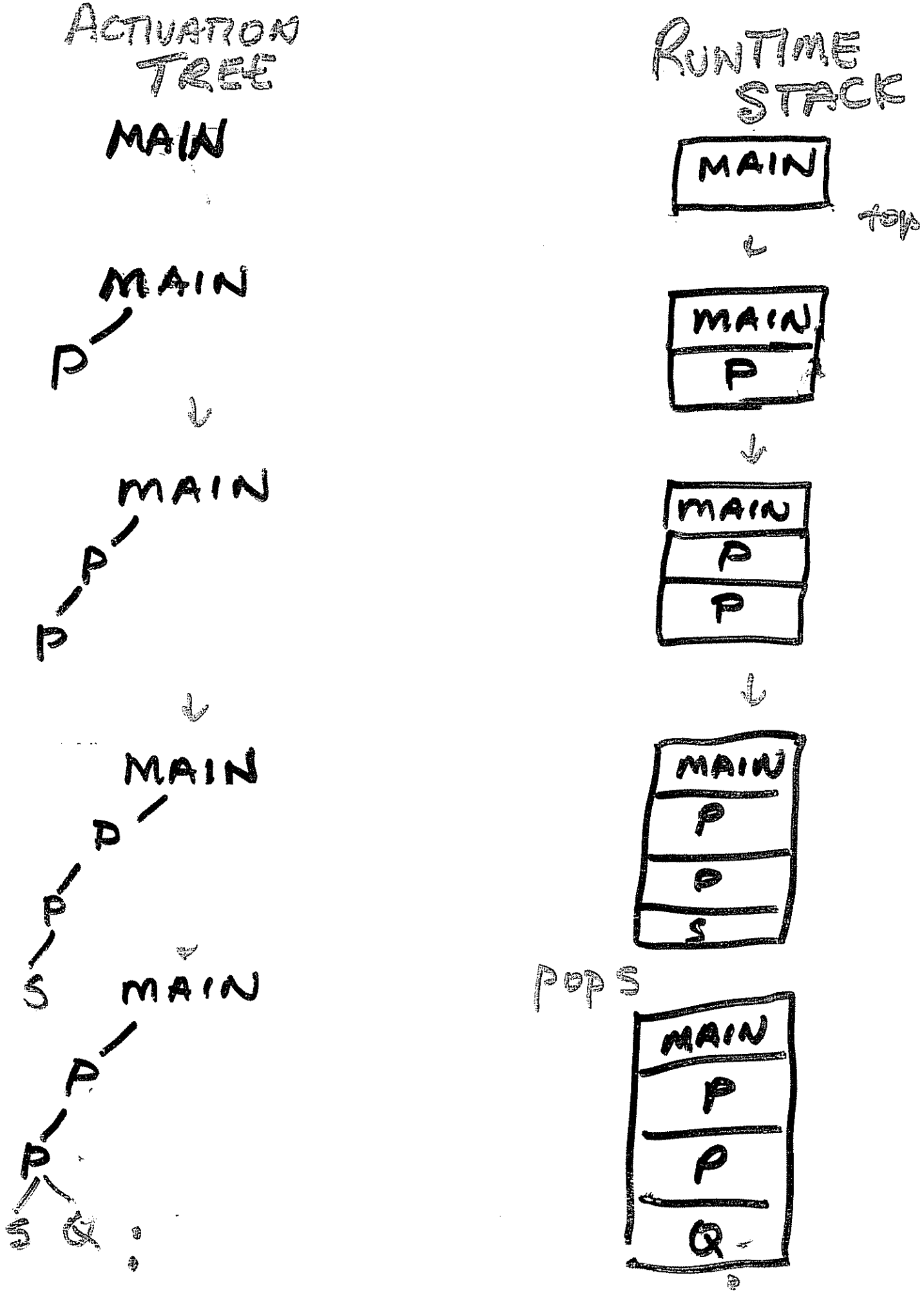


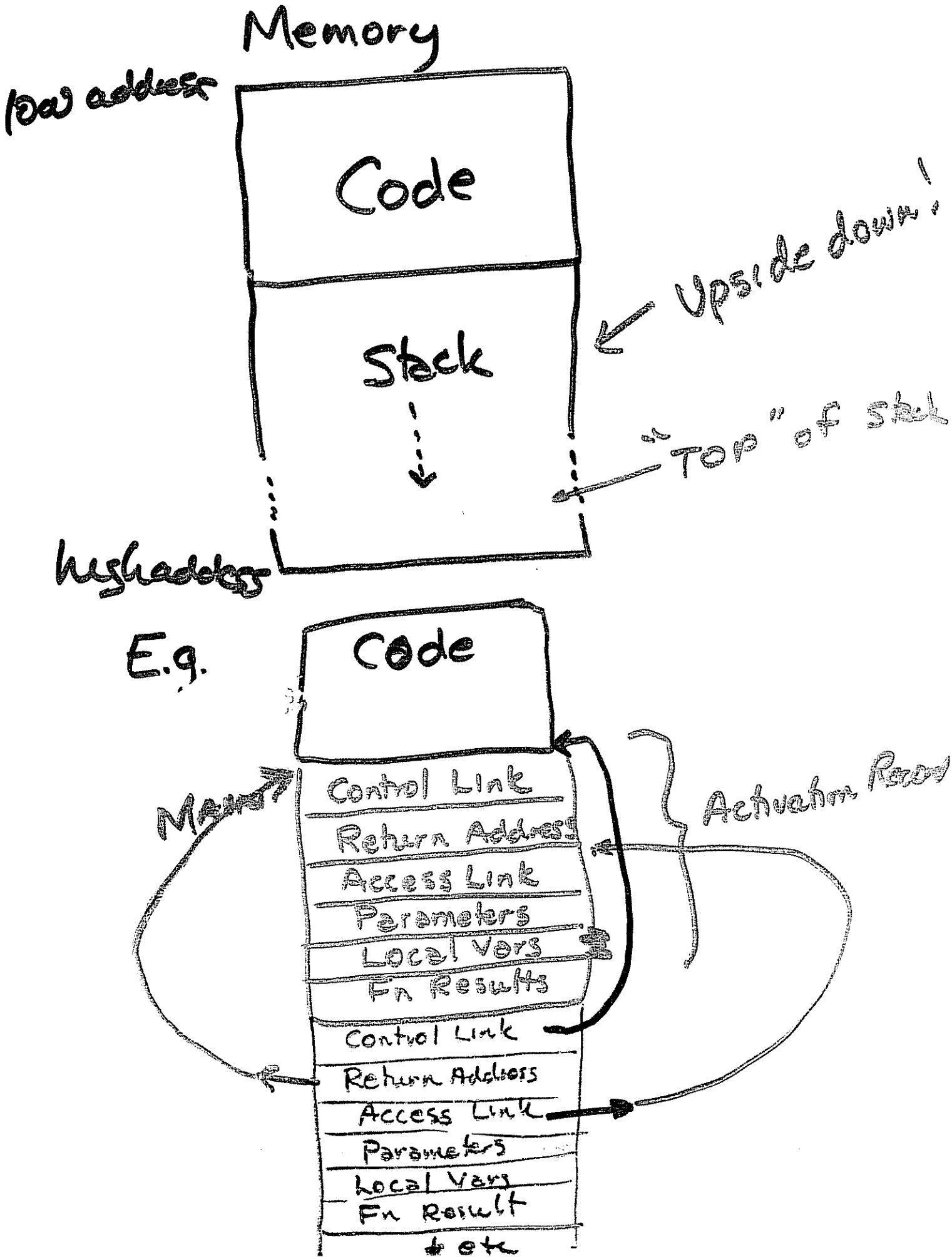
Sample Activation Trees





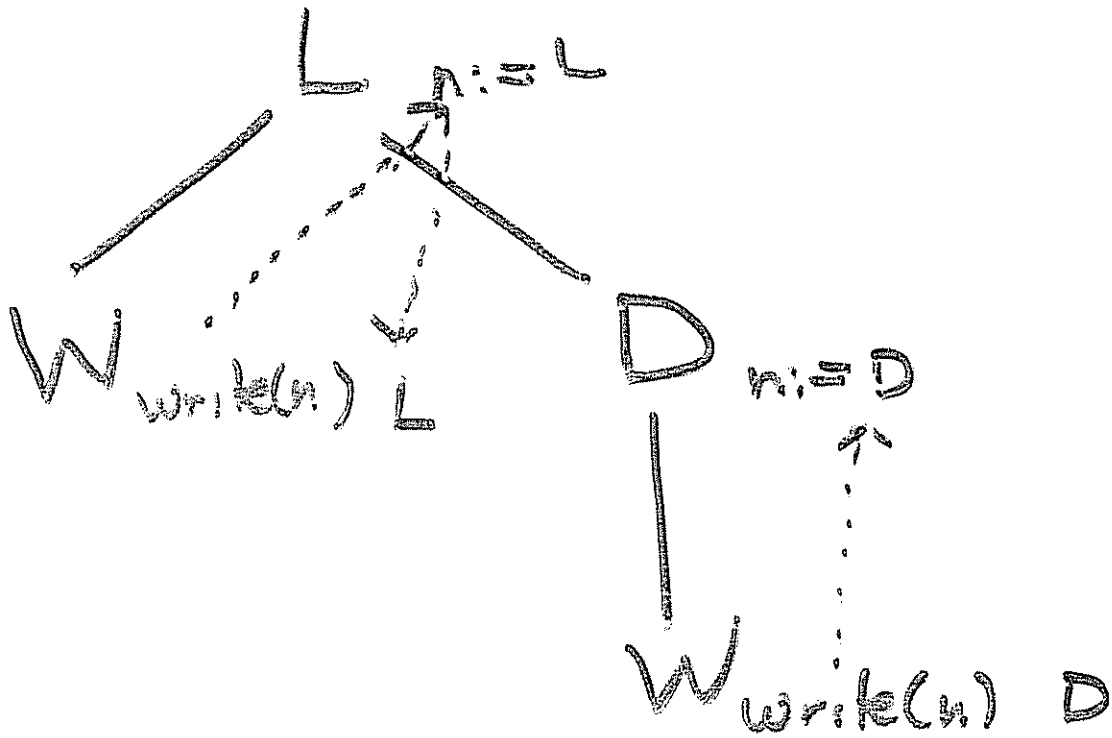



ACTIVATION TREE
MAIN
MAIN
P
MAIN
P P
MAIN
P P S
MAIN
P P Q
RUNTIME STACK
MAIN
top
MAIN
P
MAIN
P
P
MAIN
P
P
S
POP S
MAIN
P
P
Q


Memory
low address
Code
Stack
Upside down!
"Top" of stack
high address
E.g.
Code
Control Link
Return Address
Access Link
Parameters
Local Vars
Fn Results
Activation Record
Control Link
Return Address
Access Link
Parameters
Local Vars
Fn Result
etc

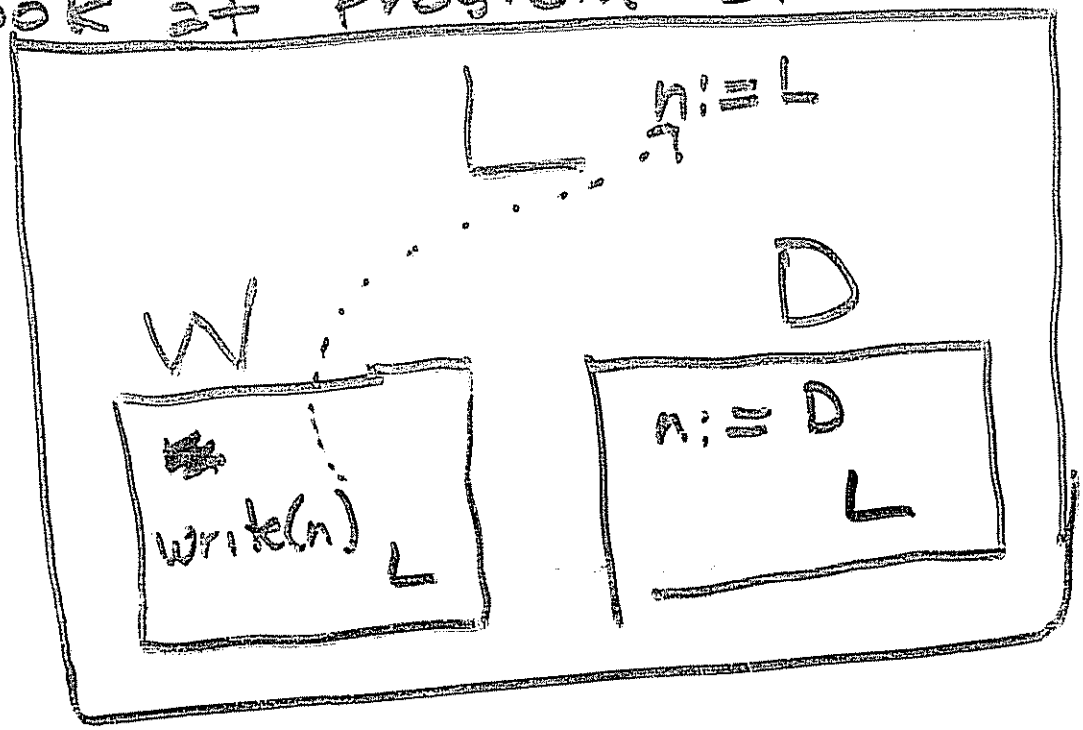
Dynamic Scope

⇒ Look @ Activation Tree

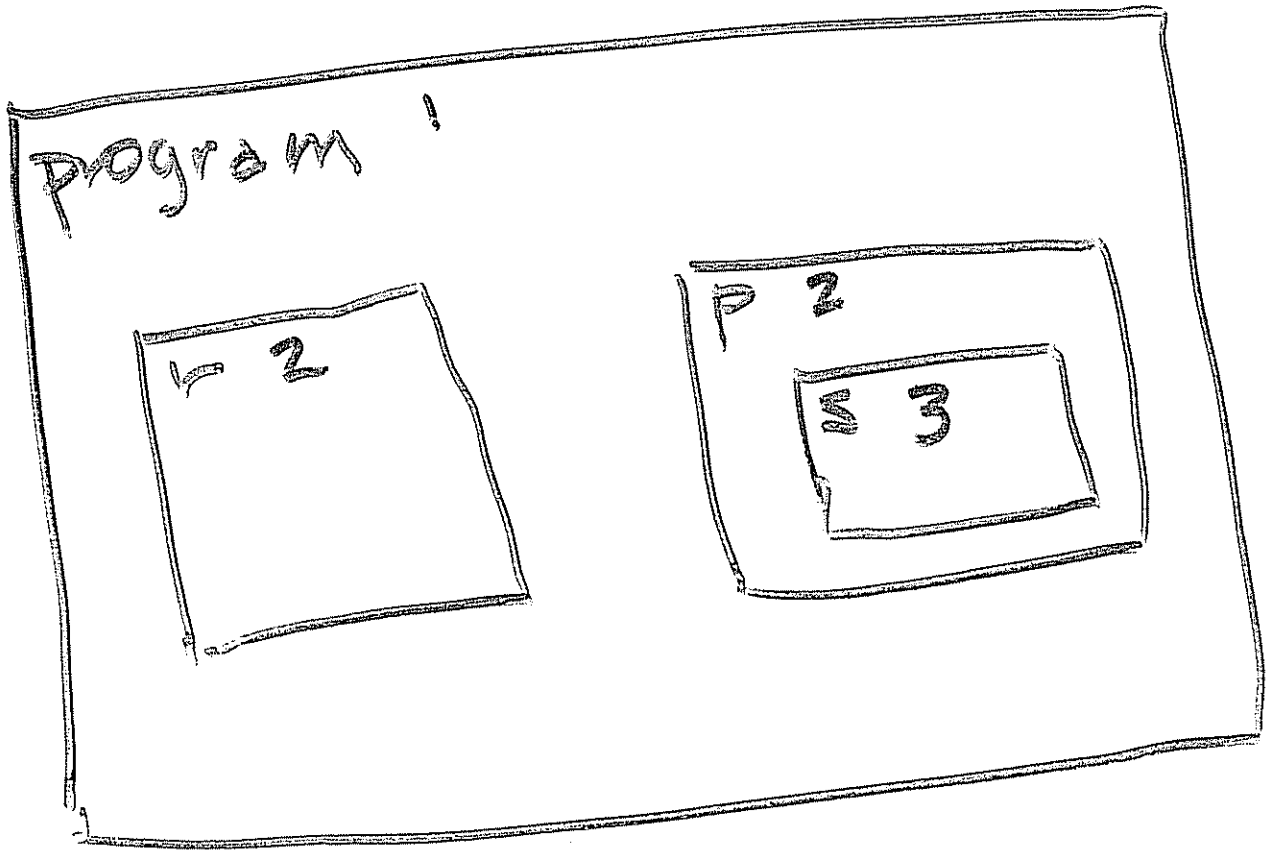


Static/Lexical Scope

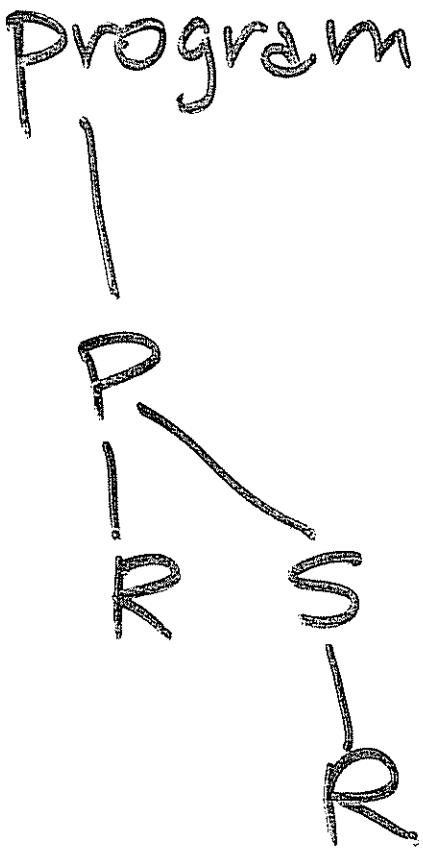
⇒ Look at Program structure



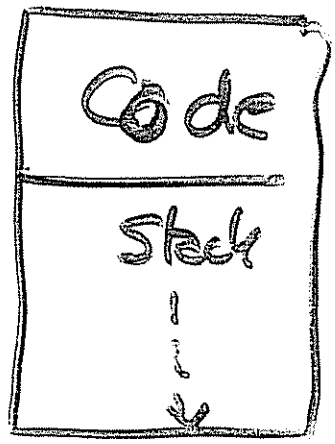
Draw the Program Structure

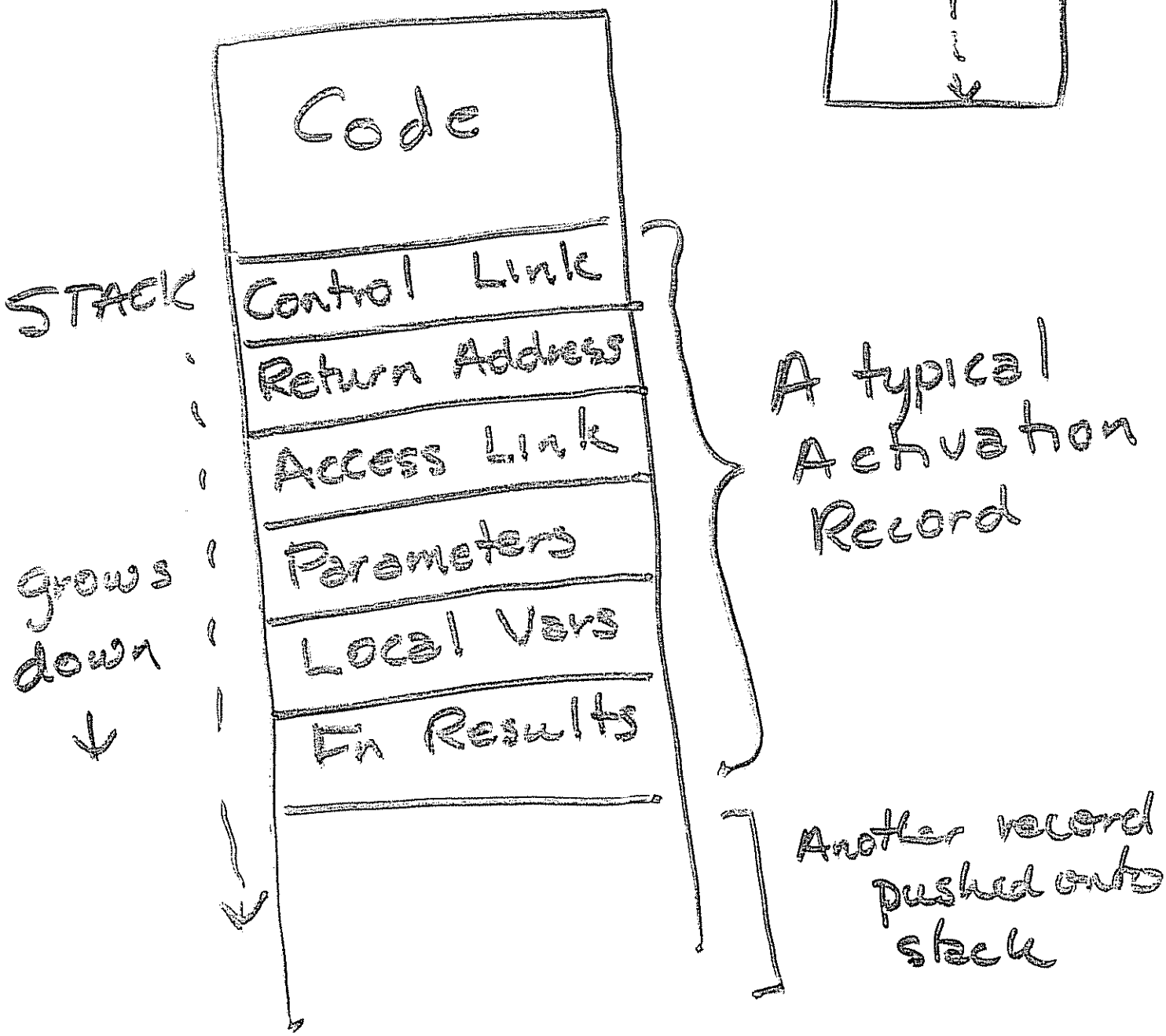


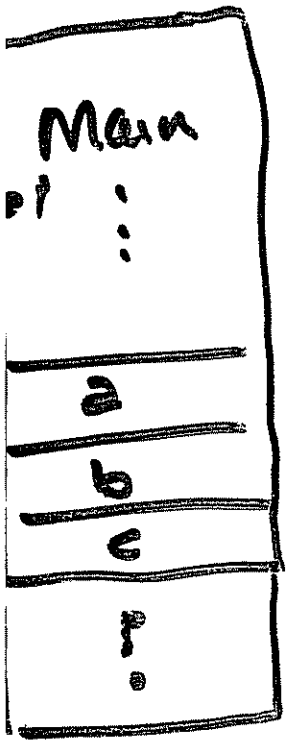
What's the Activation Tree?

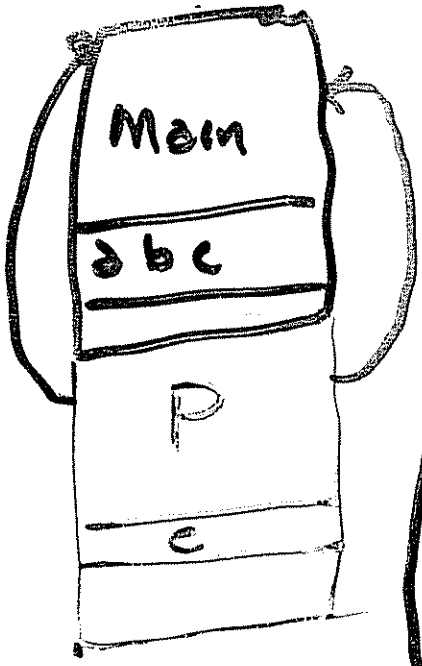


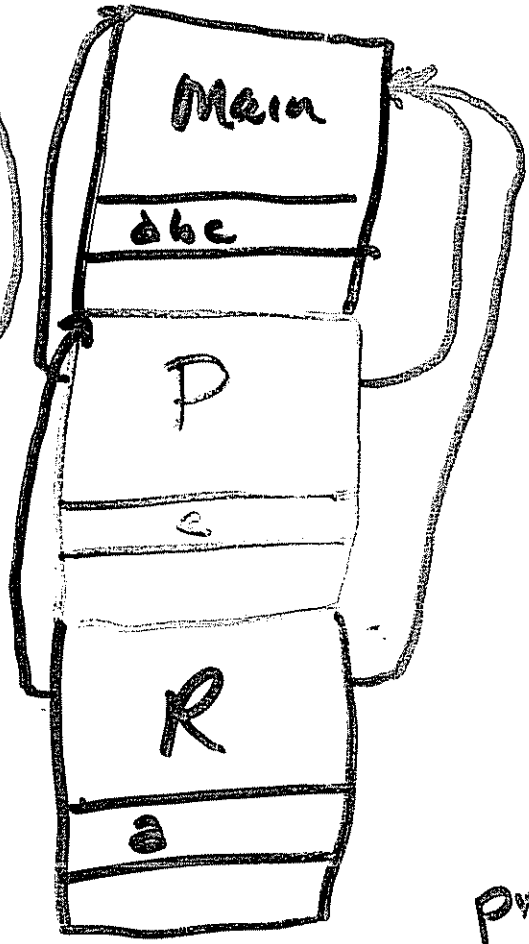
Recall What's in Our Stack!

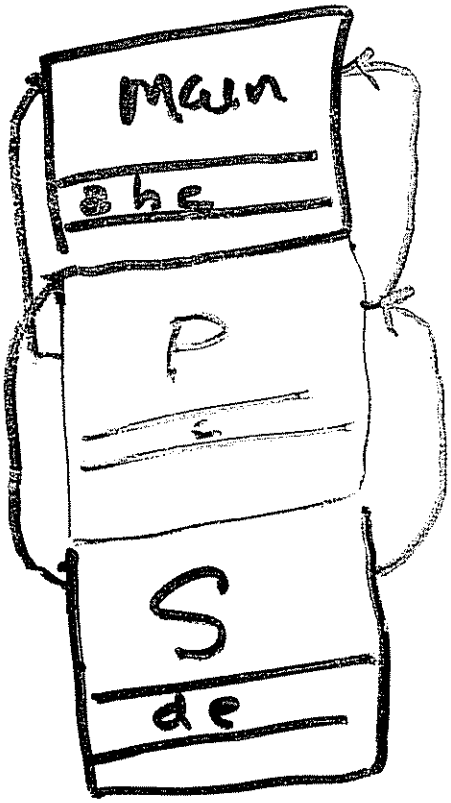


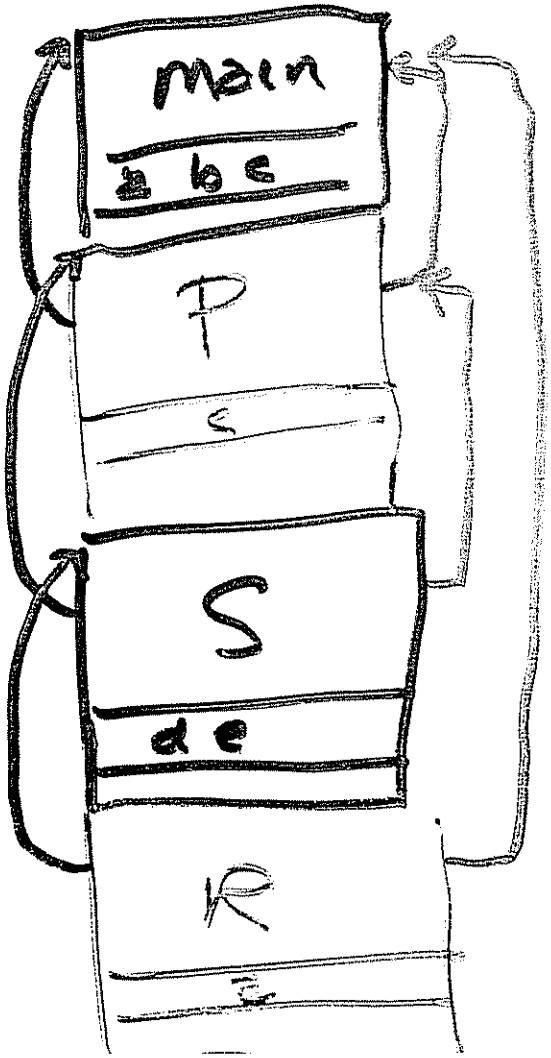


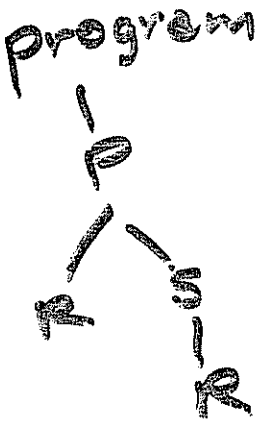

Main
a
b
c

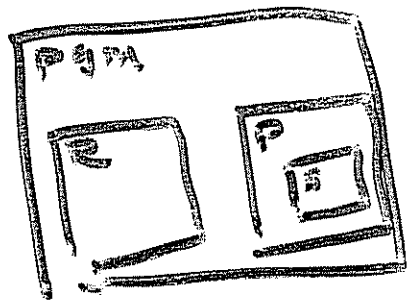

Main
a b c
P
c


Main
a b c
P
c
R
a


Main
a b c
P
c
S
d e


main
a b c
P
c
S
d e
R
e


Program
P
R
S
R


P
R
P

To determine the access link for name a, b, c in S follow n-m access links from procedure S (where a, b, c are used)

① Nesting Depth, n, of S

for a n = 3
for b n = 3
for c n = 3

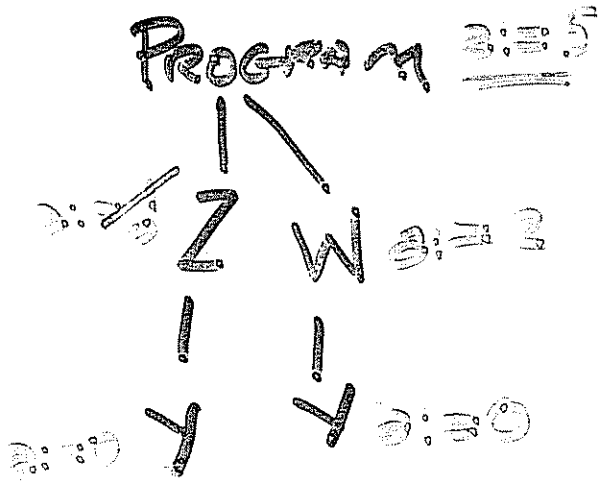
② Nesting Depth of declaration, m, of a, b, c

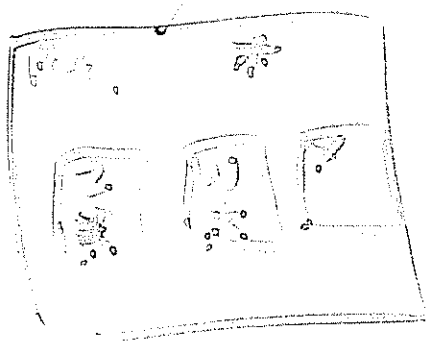
for a m = 1
for b m = 1
for c m = 2 ← because re declared in procedure P

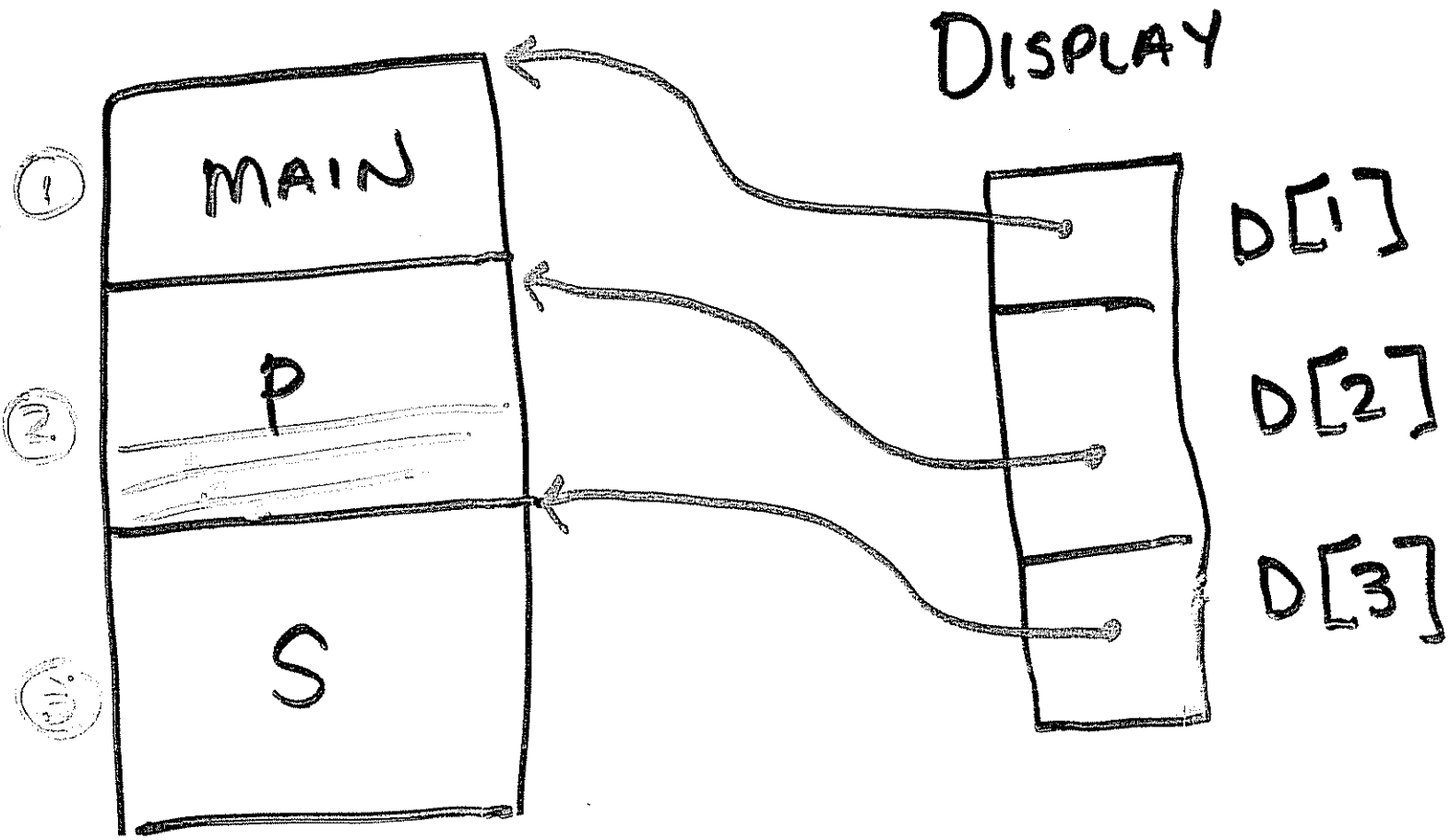
∴ For a, b follow n-m = 2 links
For c follow n-m = 1 link

Dynamic Scope Example

```
program
  a : integer;
  procedure z
    a : integer; ...
    a := 1;
    y;
    output a;
  end z;
  procedure w
    a : integer; ...
    a := 2;
    y;
    output a;
  end w;
  procedure y ...
    a := 0;
  end y;
  a := 5;
  z;
  w;
  output a;
end
```






DISPLAY
MAIN
P
S
1
2
3
D[1]
D[2]
D[3]