1) Exercise 3-5:

Try to define a procedure Sum(g, y), which has two parameters g, y. We know in ALGOL, by default, the parameters are passed by name, that means the real parameters just SUBSTITUTE the parameters in the procedure.

```
*** Procedure Sum
        real procedure sum (g, y); real g; real y;
                begin real S; S=0;
                        y:=0;
                        for y:= y + 0.01 while y<=1 do
                                S := S + g;
                        sum := S/100
                end;

*** Procedure f(x)
        real procedure f(x);
                value x; real x;
                f := x^2 +1;

*** Caller of Sum(f(x))
        begin real sumf; real x;
                sumf := sum (f(x), x)
        end
```

2) Exercise 10:

a) both x and y are passed by value:

Statements	i	A[1]	A[2]	A[3]	x	y	Cumulative output
A[1]:=7; A[2]:=11; A[3]:=13; i:=1;	1	7	11	13	?	?	
P(A[i],i);	1	7	11	13	7	1	
y:=2;	1	7	11	13	7	2	
Print(x);	1	7	11	13	7	2	7
i:=3;	3	7	11	13	7	2	
Print(x);	3	7	11	13	7	2	7, 7
i:=3;	3	7	11	13	7	2	
Print(x);	3	7	11	13	7	2	7, 7, 7
Print(y);	3	7	11	13	7	2	7, 7, 7, 2

P(i,A[i]);	3	7	11	13	3	13	
y:=2;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 7, 7, 2, 3
i:=3;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 7, 7, 2, 3, 3
i:=3;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 7, 7, 2, 3, 3, 3
Print(y);	3	7	11	13	3	2	7, 7, 7, 2, 3, 3, 3, 2

b) x is passed value and y is passed by name:

Statements	i	A[1]	A[2]	A[3]	x	y	Cumulative output
A[1]:=7; A[2]:=11; A[3]:=13; i:=1;	1	7	11	13	?	?	
P(A[i],i);	1	7	11	13	7	1	
y:=2;	2	7	11	13	7	2	
Print(x);	2	7	11	13	7	2	7
i:=3;	3	7	11	13	7	3	
Print(x);	3	7	11	13	7	3	7, 7
i:=3;	3	7	11	13	7	3	
Print(x);	3	7	11	13	7	3	7, 7, 7
Print(y);	3	7	11	13	7	3	7, 7, 7, 3
P(i,A[i]);	3	7	11	13	3	13	
y:=2;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	7, 7, 7, 3, 3
i:=3;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	7, 7, 7, 3, 3, 3
i:=3;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	7, 7, 7, 3, 3, 3, 3
Print(y);	3	7	11	2	3	2	7, 7, 7, 3, 3, 3, 3, 2

c) x is passed name and y is passed by value:

Statements	i	A[1]	A[2]	A[3]	x	y	Cumulative output
A[1]:=7; A[2]:=11; A[3]:=13; i:=1;	1	7	11	13	?	?	
P(A[i],i);	1	7	11	13	7	1	
y:=2;	1	7	11	13	7	2	
Print(x);	1	7	11	13	7	2	7
i:=3;	3	7	11	13	13	2	
Print(x);	3	7	11	13	13	2	7, 13
i:=3;	3	7	11	13	13	2	
Print(x);	3	7	11	13	13	2	7, 13, 13

Print(y);	3	7	11	13	13	2	7, 13, 13, 2
P(i,A[i]);	3	7	11	13	3	13	
y:=2;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 13, 13, 2, 3
i:=3;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 13, 13, 2, 3, 3
i:=3;	3	7	11	13	3	2	
Print(x);	3	7	11	13	3	2	7, 13, 13, 2, 3, 3, 3
Print(y);	3	7	11	13	3	2	7, 13, 13, 2, 3, 3, 3, 2

d) both x and y are passed by name:

Statements	i	A[1]	A[2]	A[3]	x	y	Cumulative output
A[1]:=7; A[2]:=11; A[3]:=13; i:=1;	1	7	11	13	?	?	
P(A[i],i);	1	7	11	13	7	1	
y:=2;	2	7	11	13	11	2	
Print(x);	2	7	11	13	11	2	11
i:=3;	3	7	11	13	13	3	
Print(x);	3	7	11	13	13	3	11, 13
i:=3;	3	7	11	13	13	3	
Print(x);	3	7	11	13	13	3	11, 13, 13
Print(y);	3	7	11	13	13	3	11, 13, 13, 3
P(i,A[i]);	3	7	11	13	3	13	
y:=2;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	11, 13, 13, 3, 3
i:=3;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	11, 13, 13, 3, 3, 3
i:=3;	3	7	11	2	3	2	
Print(x);	3	7	11	2	3	2	11, 13, 13, 3, 3, 3, 3
Print(y);	3	7	11	2	3	2	11, 13, 13, 3, 3, 3, 3, 2

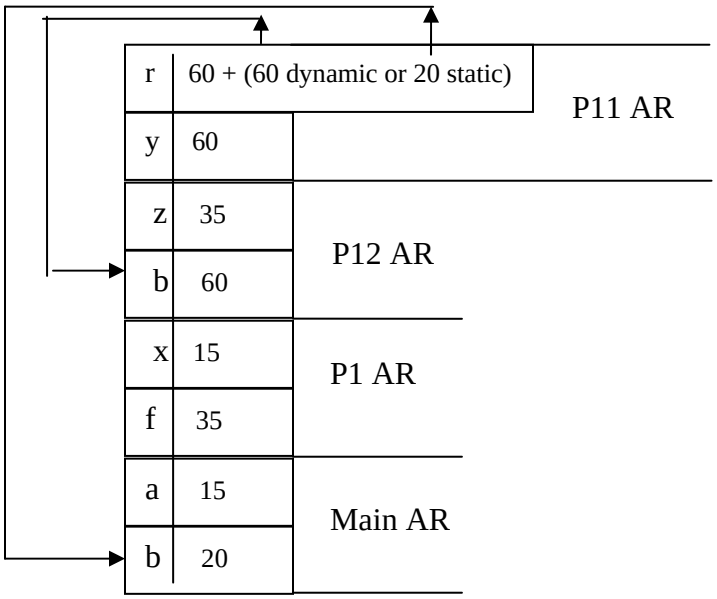
e) both x and y are passed by value-result:

Statements	i	A[1]	A[2]	A[3]	x	y	Cumulative output
A[1]:=7; A[2]:=11; A[3]:=13; i:=1;	1	7	11	13	?	?	
P(A[i],i);	1	7	11	13	7	1	
y:=2;	1	7	11	13	7	2	
Print(x);	1	7	11	13	7	2	7
i:=3;	3	7	11	13	7	2	

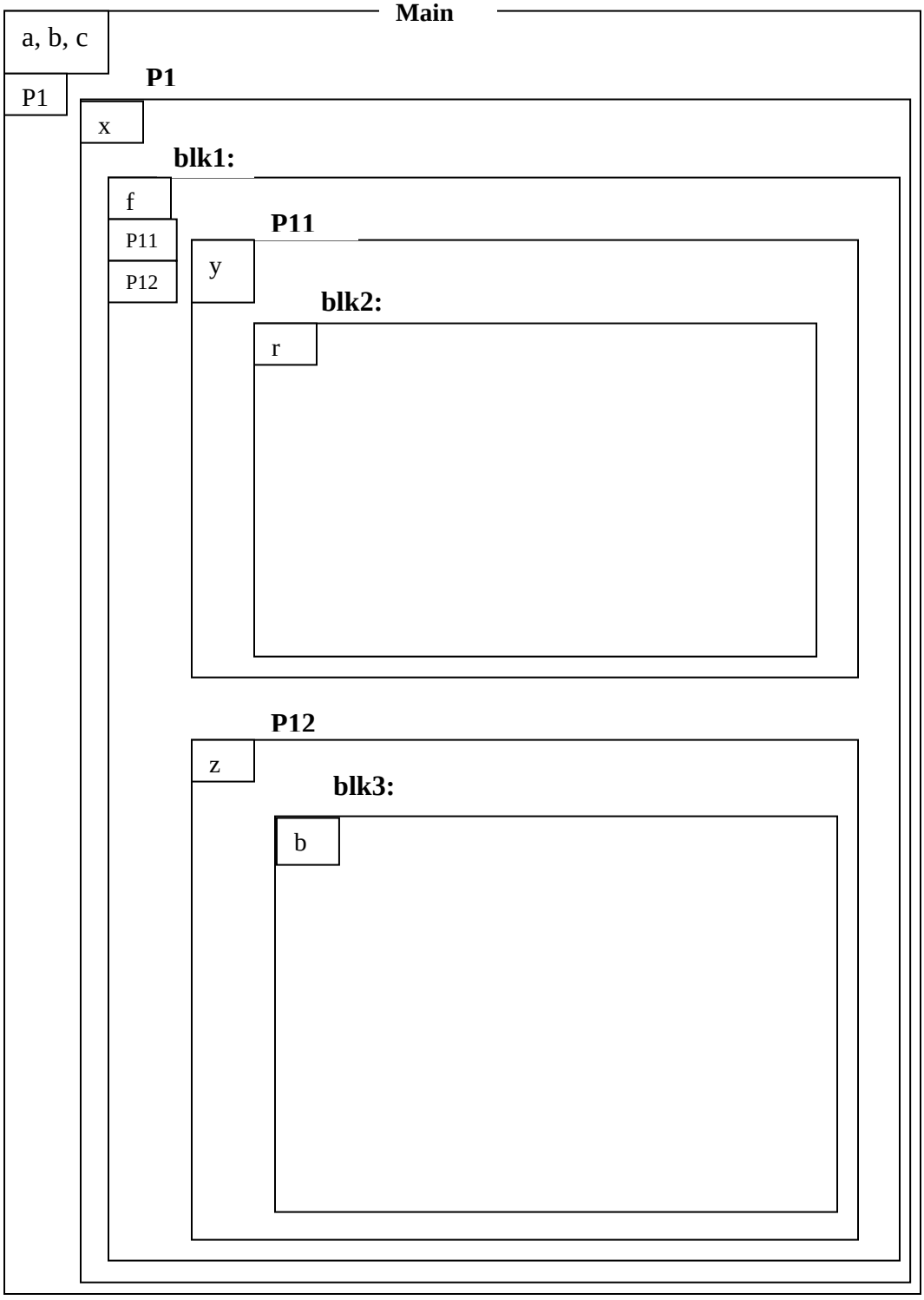
Print(x);	3	7	11	13	7	2	7, 7
i:=3;	3	7	11	13	7	2	
Print(x);	3	7	11	13	7	2	7, 7, 7
Print(y);	3	7	11	13	7	2	7, 7, 7, 2
{return}	2	7	11	13	7	2	
P(i,A[i]);	2	7	11	13	2	11	
y:=2;	2	7	11	13	2	2	
Print(x);	2	7	11	13	2	2	7, 7, 7, 2, 2
i:=3;	3	7	11	13	2	2	
Print(x);	3	7	11	13	2	2	7, 7, 7, 2, 2, 2
i:=3;	3	7	11	13	2	2	
Print(x);	3	7	11	13	2	2	7, 7, 7, 2, 2, 2, 2
Print(y);	3	7	11	13	2	2	7, 7, 7, 2, 2, 2, 2, 2
{return}	2	7	2	13	2	2	

3) a) i) static: r = 80 ii) dynamic: r = 120

The picture of the run-time stack when printing “r”:



b)



i) No ii) Yes iii) No way, blocks are not to be called

c) i) from Q1 we can call: P1, Q1
from Q12 we can call Q11, Q1, P1, Q12
from P11 we can call P12, P1, Q1, P11
from P1 we can call: Q1, P1

ii) No, there is no use of the contour diagram in the case of dynamic scoping, and the compiler (and us) will never be able to answer the above questions about visibility until run-time.

d) The dynamic run-time scenario that allows for calling P11 from the code of Q12 is following sequence of steps:
1- block "main" calls P1,
2- P1 calls Q1,
3- Q1 calls Q11,
4- Q11 now can call P12 since if we follow the dynamic links, we can get to the definition of P12 in the AR of P1; all ARs are stacked according the above calling sequence.

EXTRA CREDIT:

Yes, dynamic scoping is a violation of abstraction since it allows the penetration of outsider "callee" procedure to its "caller" internal scope (presumably hidden), and all of its callers in the dynamic chain of DLs, for inclusion as part of their environments. When the programmer nest (hide) scopes he/she means abstraction.