SIMPLESEM

Esercizio 1

Si consideri il seguente frammento di programma scritto in un ipotetico linguaggio simile al C:

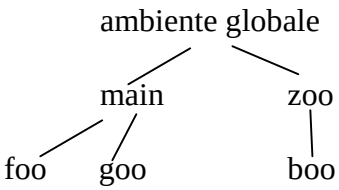
```
int x = 1, y = 3;

main() {
  int x = 2;

  goo() {
    x++;     (2)
    zoo();
  }

  foo() {
    int x=1;
    goo();
  }

  x=y+2;
  foo();
  y=x-1;
}            (3)

zoo() {
  int y=1;

  boo() {
    x = 2;
    y++;     (1)
  }

  x=7 ;
  boo();
}
```

(a) Si assumano regole di scoping e di tipizzazione entrambe statiche. Mostrare lo stato dell'interprete SIMPLESEM nei punti indicati con (1), (2), e (3) con il contenuto delle variabili.
(b) Si assuma ora che il linguaggio abbia regole di tipizzazione e di scoping entrambe dinamiche. Si discuta come cambia la semantica del programma, e in particolare i valori delle variabili nei punti sopra indicati.

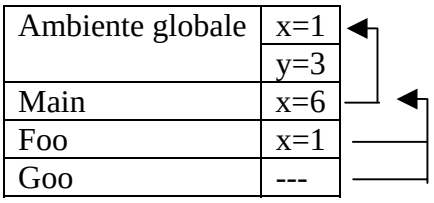
Soluzione

(a) – regole di visibilità statiche (indichiamo tramite frecce solamente i link statici)

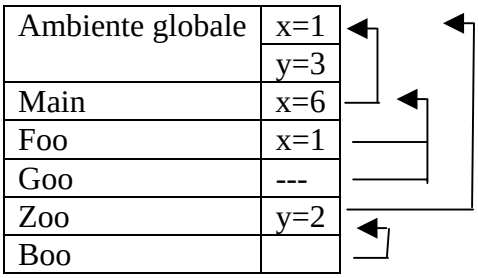
Albero degli annidamenti statici:



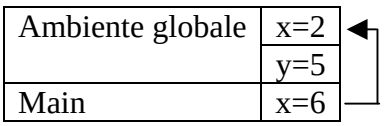
Punto 2:



Punto 1:



Punto 3:



CASO (b): regole di visibilità dinamiche (indichiamo tramite frecce i link dinamici)

Punto 2:

Ambiente globale	x=1
	y=3
Main	x=5
Foo	x=2
Goo	---

Punto 1:

Ambiente globale	x=1
	y=3
Main	x=5
Foo	x=2
Goo	---
Zoo	y=2
Boo	

Punto 3:

Ambiente globale	x=1
	y=4
Main	x=5

Esercizio 2
Si consideri il seguente frammento di programma:

```
program Esame
var x,y,z,w:integer;
        procedure P1()
        var a,b,c:integer;
        ...
        end P1;
        procedure P2()
        var a,d,e: integer;
                procedure P3()
                var f,g:integer;
                z=a+x+g+f+w;          (*)
                …
                end P3;
        end P2;
…
end Esame;
```

1. Mostrare lo stato della macchina astratta dopo la seguente catena di chiamate
   Esame $\rightarrow$ P2 $\rightarrow$ P1 $\rightarrow$ P2 $\rightarrow$ P3
2. Illustrare come staticamente vengono tradotte in coppie <distanza, offset> le variabili presenti nella istruzione contrassegnata da (*)

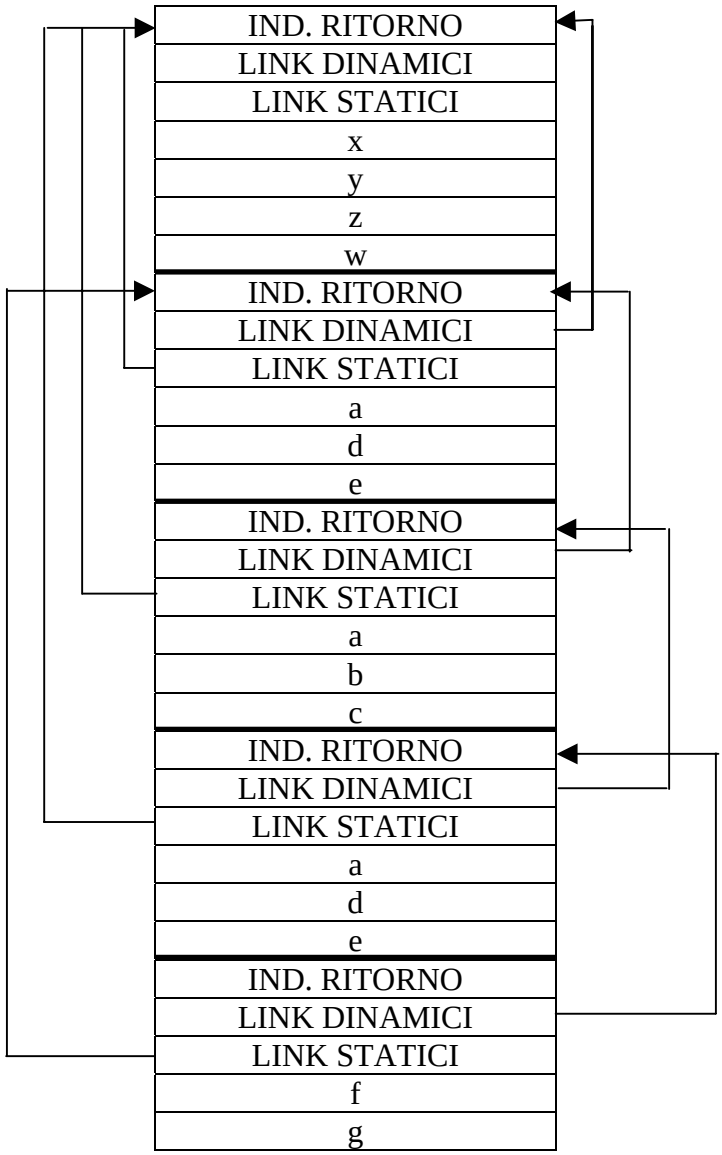
Esami:

P2:

P1:

P2:

P3:



z=a+x+g+f+w
z=<2,5>
a=<1,3>
x=<2,3>
g=<0,4>
f=<0,3>
w=<2,6>

Si dica, giustificando brevemente la risposta, se la seguente affermazione è vera o falsa:

Se un programma C non fa uso né di blocchi né di sottoprogrammi annidati (ossia se consiste solo del main e di una lista di sottoprogrammi dichiarati tutti allo stesso livello del main) la semantica determinata dalla catena statica equivale alla semantica della catena dinamica.

Risposta:

Falsa. Infatti se si adotta la regola della catena statica le uniche variabili non locali usabili dai vari sottoprogrammi sono quelle globali; si invece si adotta la regola della catena dinamica un generico sottoprogramma P "cercherà" una variabile non locale x nel record di attivazione del sottoprogramma Q che lo ha chiamato e via via più in basso nella pila.

Esercizio 3

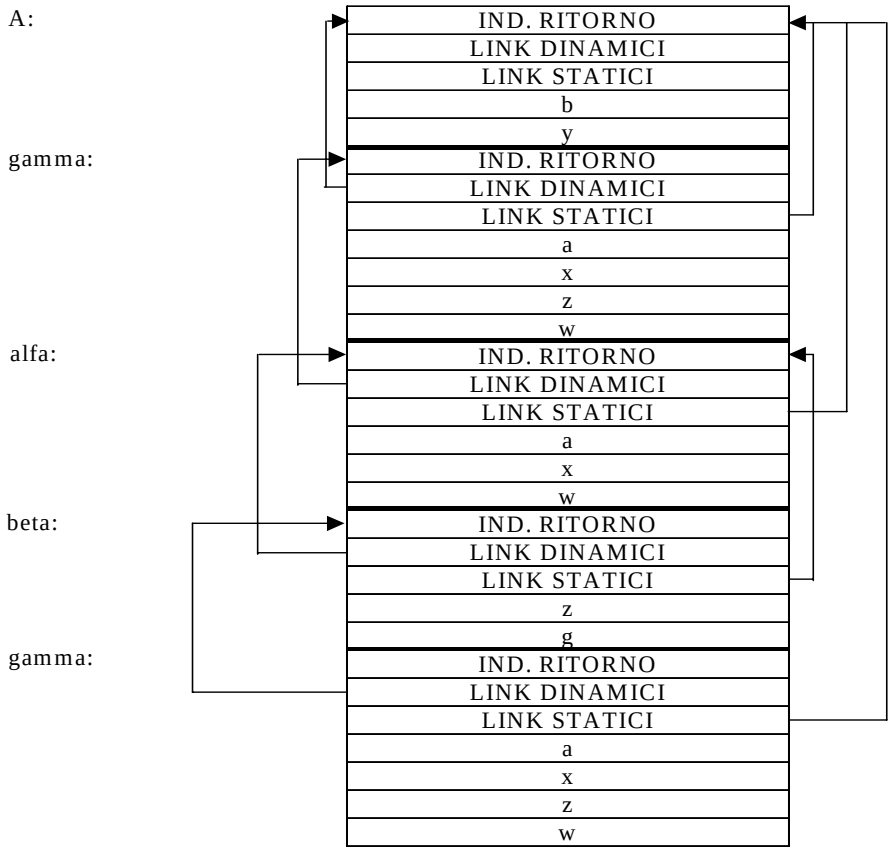
```
1.program A{
2.      integer b,y;
3.      routine alfa() {
4.              integer a,x,w;
5.              routine beta () {
6.                      integer z,g;
7.                      a=z+x+b+w;
8.              };…….
9.              x=a+y; …….
10.     };
11.     routine gamma() {
12.             integer a,x,z,w; …….
13.      };
14.}
```

Considerata la seguente catena di chiamate:
A–>gamma –>alfa –>beta –>gamma

schizzare lo stato della macchina astratta
–link statici
–link dinamici

Regole di scope statico

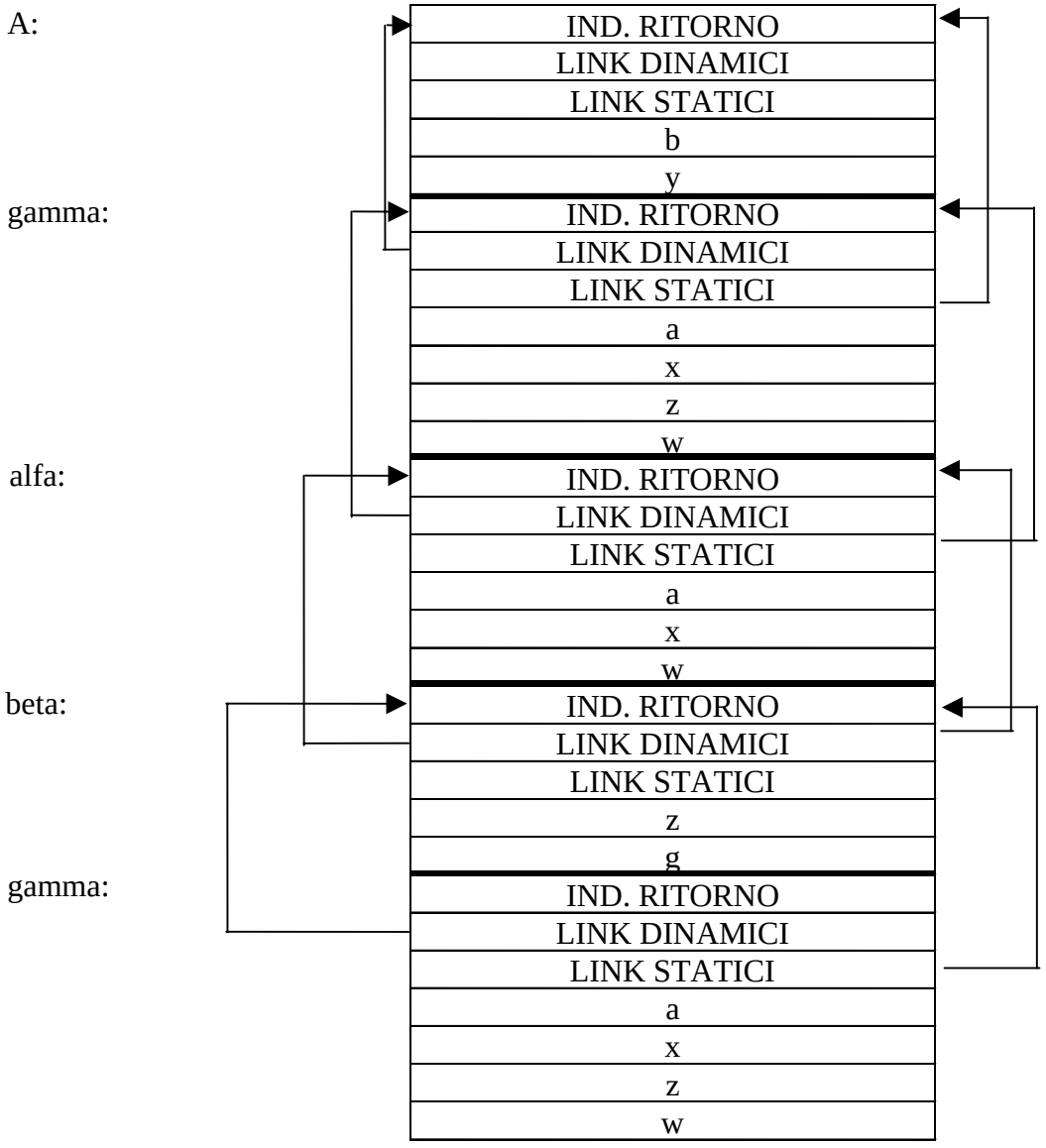
•Linea 7: a=z+x+b+w;
a=<1,3>
z=<0,3>
x=<1,4>
b=<2,3>
w=<1,5>
•Linea 9: x=a+y;
x=<0,4>
a=<0,3>
y=<1,4>



Regole di scope dinamico

Linea 7: a=z+x+b+w;
a=<1,3>
z=<0,3>
x=<1,4>
b=<3,3>
w=<1,5>

Linea 9: x=a+y;
x=<0,4>
a=<0,3>
y=<2,4>



Esercizio 4

Dato il seguente programma

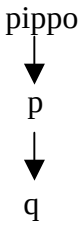
```
program pippo
var a,b,c:integer
        procedure p(procedure x)
        var a,b,d:integer;
                procedure q
                var b,e:integer;
                ..............
                end q;
        if c=0 then
                begin
                        c:=c+1;
                        x(q);
                end
        else x;
        end p;
..............
c:=0;
p(p);
end;
```

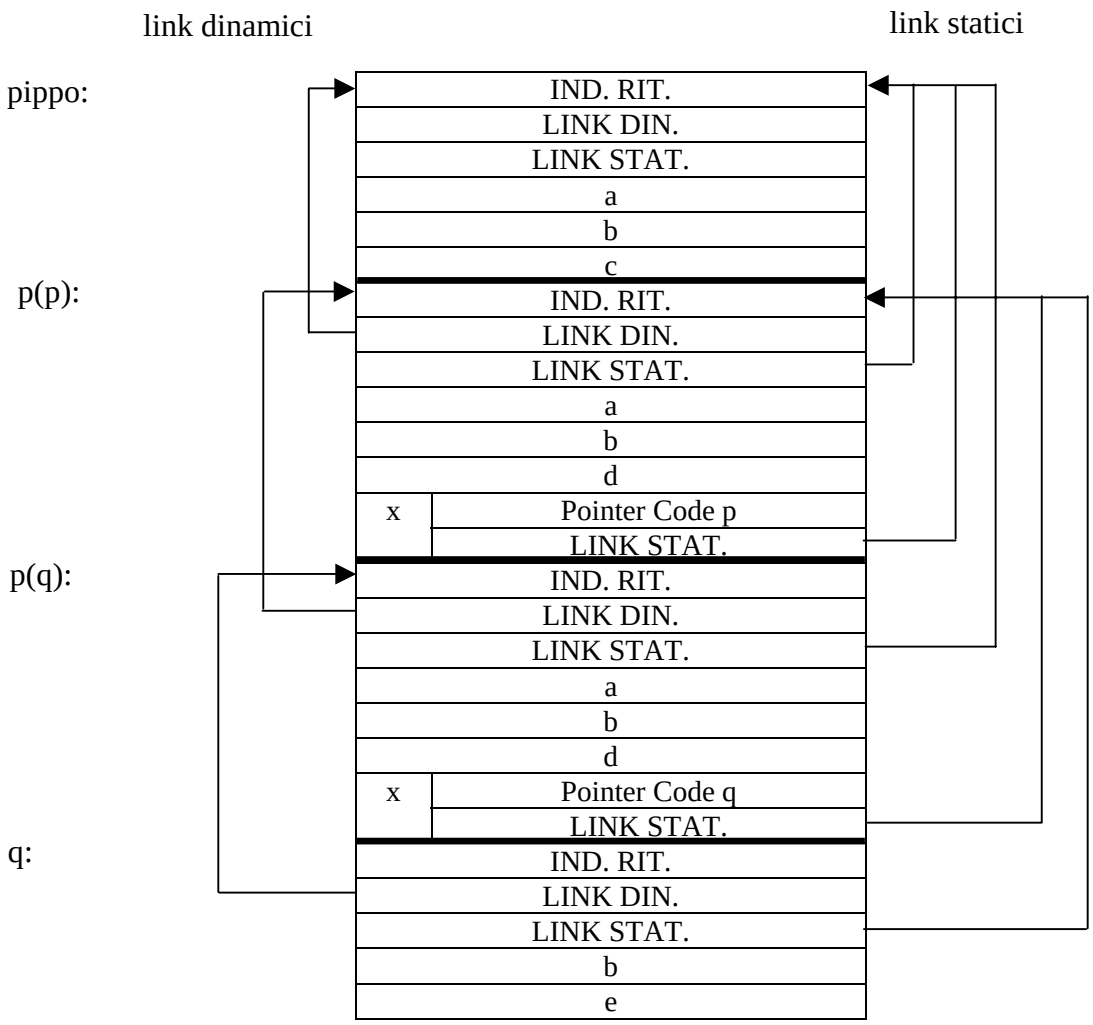
schizzare lo stato della macchina astratta, tracciando link statici e dinamici, fino a quando q viene chiamata per la prima volta.

Soluzione

Pippo pone c=0 e chiama p(p)
In p(p) si entra nel ramo then dell'if ponendo quindi c=1 e chiamando p(q)
In p(q) si entra nel ramo else dell'if quindi si chiama q
Quindi la sequenza delle chiamate e':
pippo → p(p) → p(q) → q

Annidamento statico :




link dinamici
link statici
pippo:
IND. RIT.
LINK DIN.
LINK STAT.
a
b
c
p(p):
IND. RIT.
LINK DIN.
LINK STAT.
a
b
d
x
Pointer Code p
LINK STAT.
p(q):
IND. RIT.
LINK DIN.
LINK STAT.
a
b
d
x
Pointer Code q
LINK STAT.
q:
IND. RIT.
LINK DIN.
LINK STAT.
b
e

Esercizio 5

Dato il seguente programma:

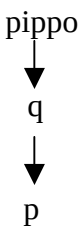
```
program pippo
var l,m,n:real;
        procedure q(procedure w)
                procedure p
                var r,s:real;
                .....
                end
        if n>0 then w
        else begin
                n:=n+1.77;
                w(p);
        end
.....
n:= -0.5;
q(q);
end
```

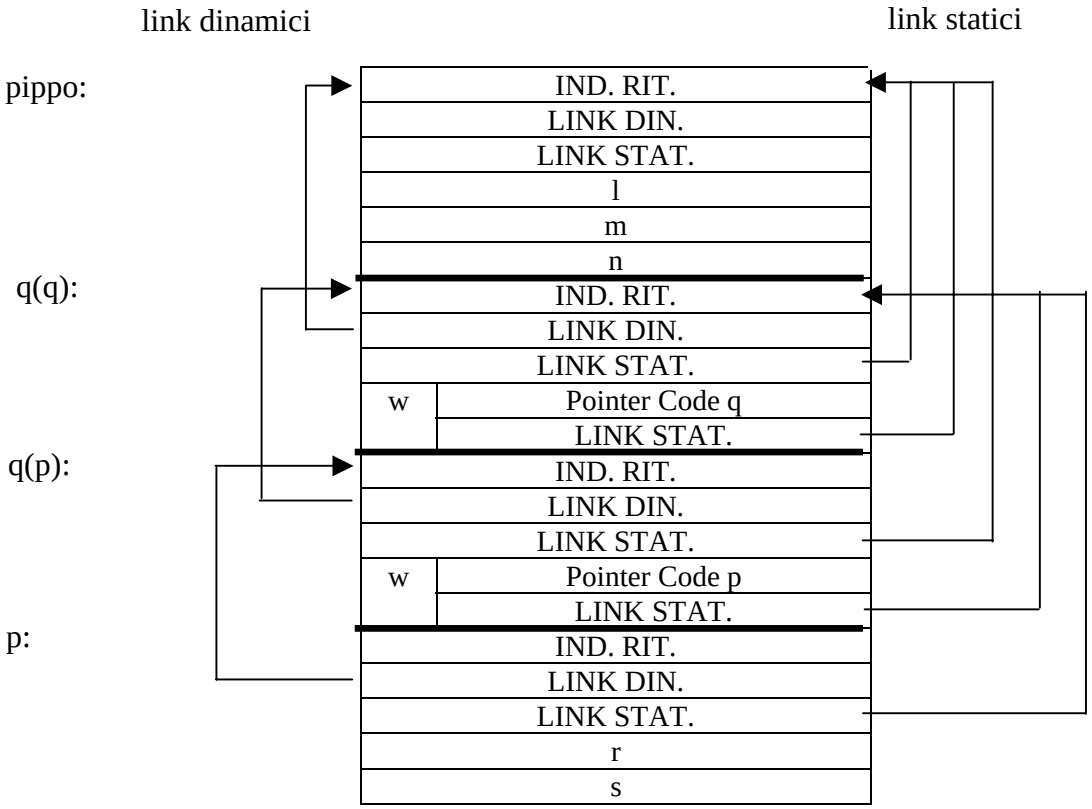
schizzare lo stato della macchina astratta, tracciando link statici e dinamici, fino alla chiamata di w effettuata nel ramo then della procedura q. Si scriva inoltre come viene tradotto l'accesso alla variabile n in termini della coppia (distanza offset) [scope statico].

Soluzione

Pippo pone n=-0.5 e chiama q(q)
In q(q) si entra nel ramo else dell'if ponendo quindi n=-0.5+1.77=1.27 e chiamando q(p)
In q(p) si entra nel ramo then dell'if quindi viene chiamata p
Quindi la sequenza delle chiamate e':
pippo $\rightarrow$ q(q) $\rightarrow$ q(p) $\rightarrow$ p

Annidamento statico :





Accesso alla variabile n da pippo: n=<0,5>
Accesso alla variabile n per la prima chiamata di q: n=<1,5>
Accesso alla variabile n per la seconda chiamata di q: n=<1,5> (valgono regole di scope statico)

Esercizio 6 (Passaggio Parametri)

```
1. program esempio;
2.  var x:integer
3.      procedure p(y:integer)
4.      begin
5.              x:=5;
6.              y:=y+x;
7.      end;
8.  begin
9.      x:=10;
10.     p(x);
11.     write(x);
12. end ;
```

By reference
(parametri formali e attuali dividono la stessa area di memoria)

# Riga	x	y
9	10	
3	10	10
5	5	5
6	10	10
11	10	

By value
(par. attuale copiato in quello formale che sostituisce la var. locale nella procedura)

# Riga	x	y
9	10	
3	10	10
5	5	10
6	5	15
11	5	

By result
(par. formale copiato in quello attuale al termine della procedura)

# Riga	x	y
9	10	
3	10	0
5	5	0
6	5	5
11	5	

By value/result

# Riga	x	y
9	10	
3	10	10
5	5	10
6	5	15
11	15	

By name
(nel testo della procedura i par. formali sono sostituiti da quelli attuali)

# Riga	x	y
9	10	
3	10	10
5	5	5
6	10	10
11	10	

Informatica 3
Prima prova intermedia, 5 Maggio 2004

ESERCIZIO 1
Si consideri il programma seguente scritto in un inesistente linguaggio di programmazione la cui sintassi e semantica sono **basate** sul C.

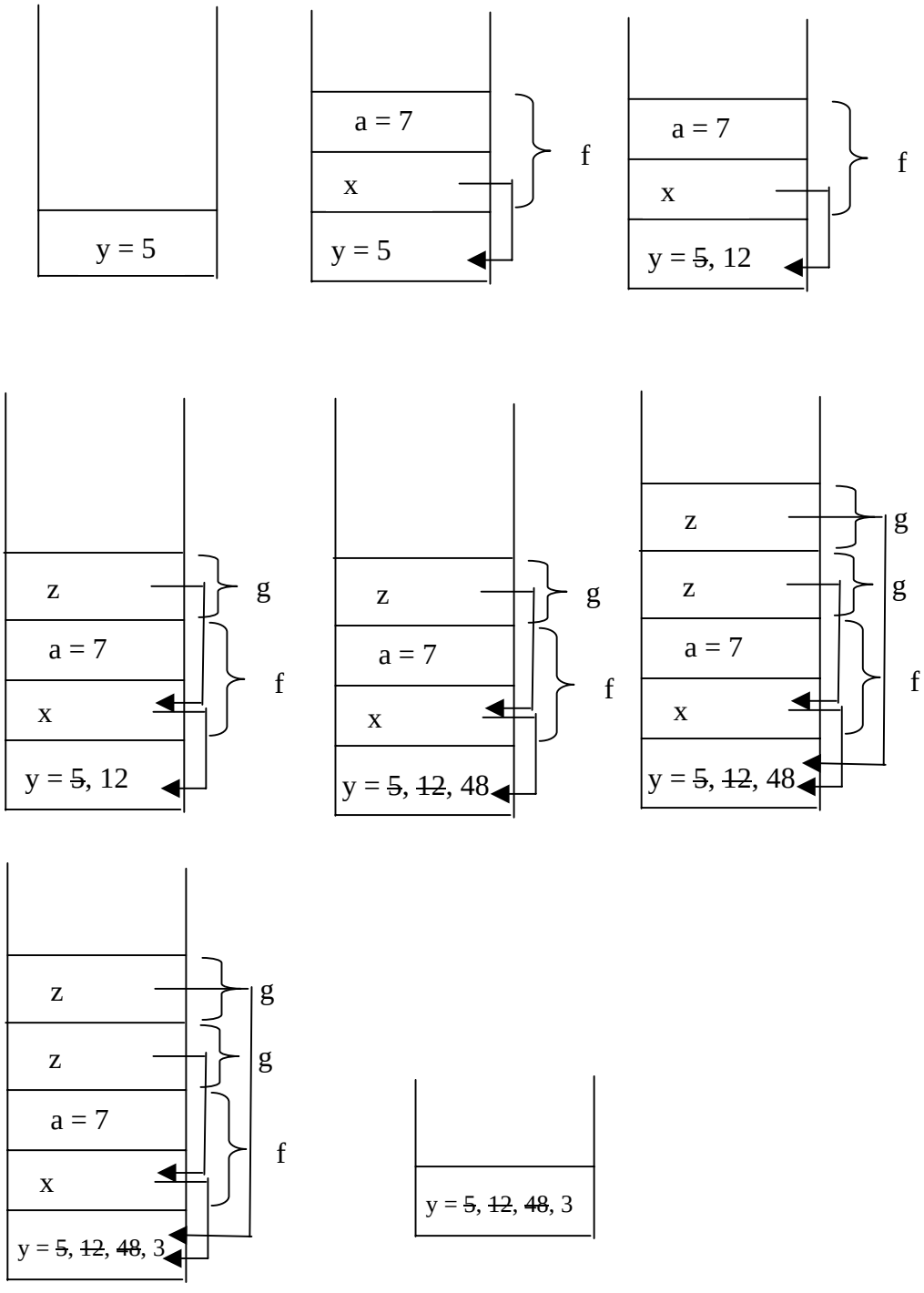
```
# include <stdio.h>
        int y;
        void f(int x)
        {
.               int a = 7;
                void g(int z)
                {
                        if (z<=12 AND z>5)
                        {
                                y=z+3*y;
                                g(y);
                        }
                        elseif  z<=5
                                z=z+2;
                        else
                                z=3;
                }
                if x<=5
                        x=x+a;
                        g(x);
                else{
                        x=a;
                        y=3;
                }
        }
void main ()
{
        y=5;
        f(y);
        printf("%d", y);
};
```

Si assuma che il programma venga eseguito applicando la regola della catena statica; si mostrino i principali passaggi e il valore stampato a video nel caso in cui il passaggio parametri avvenga

- per indirizzo
- per valore
- per risultato (con valore iniziale di default 5) ???
- per valore-risultato

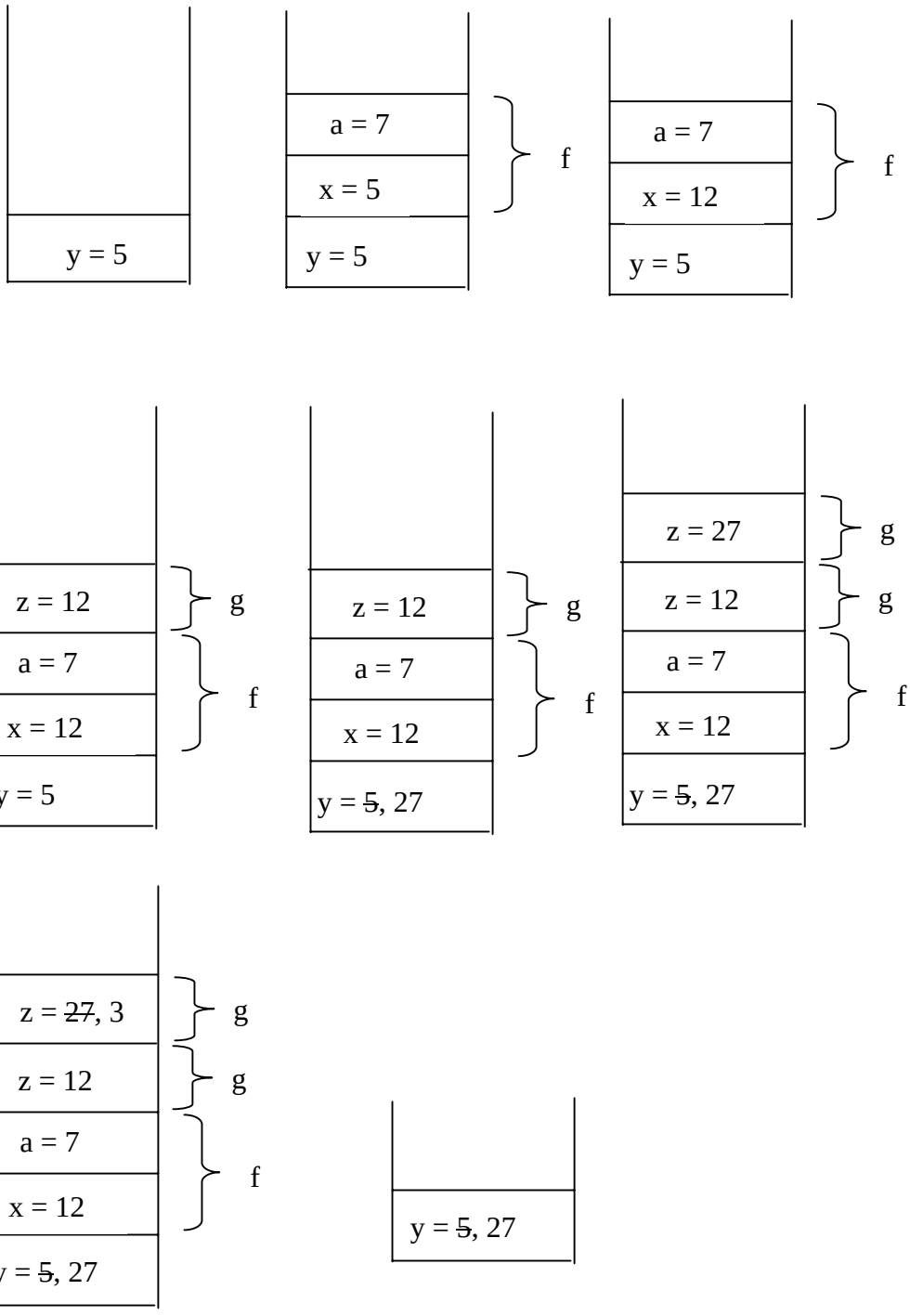
Soluzioni

Passaggio per indirizzo



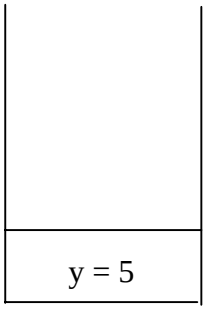
Risultato finale: y = 3

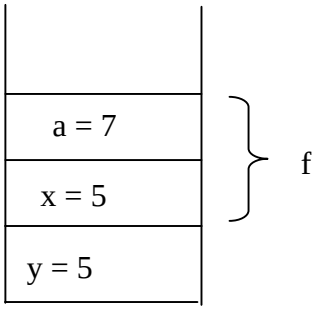
Passaggio per valore

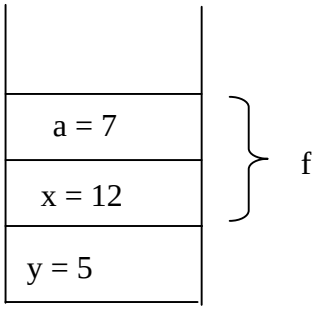


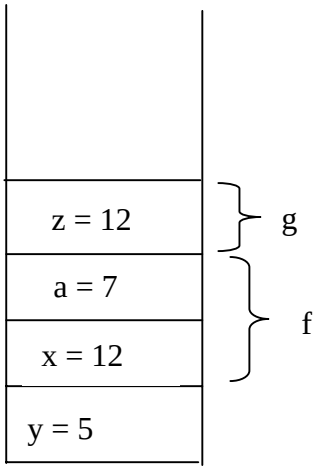
Risultato finale: y = 27

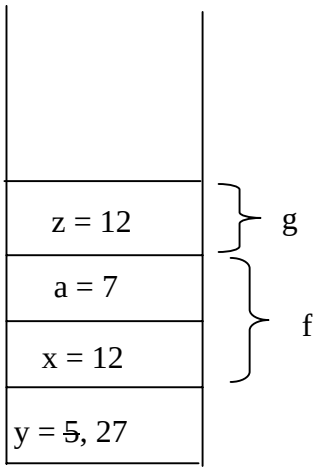
Passaggio per valore-risultato

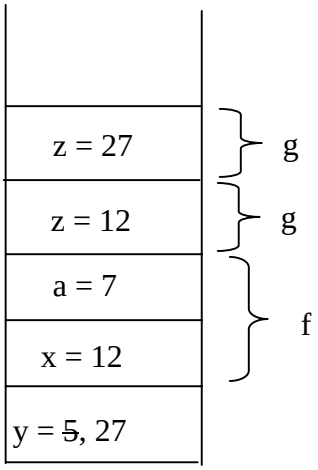


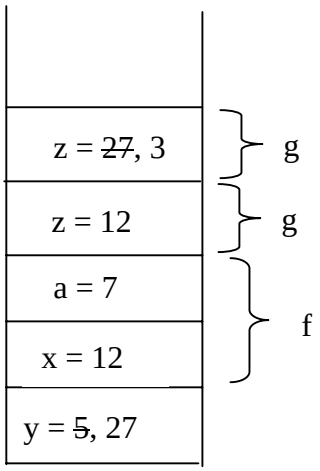


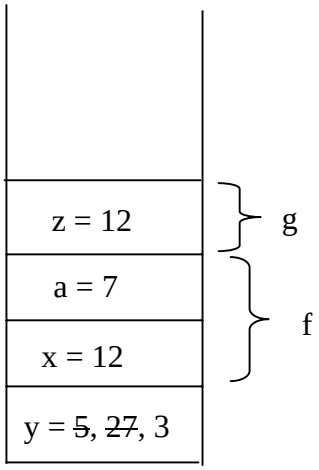


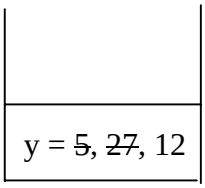






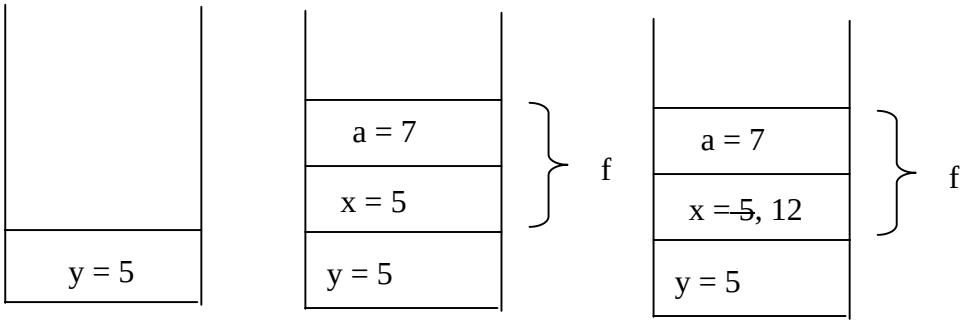


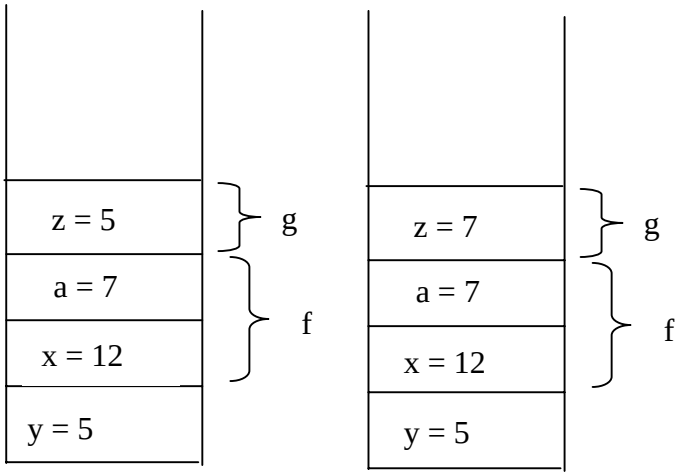


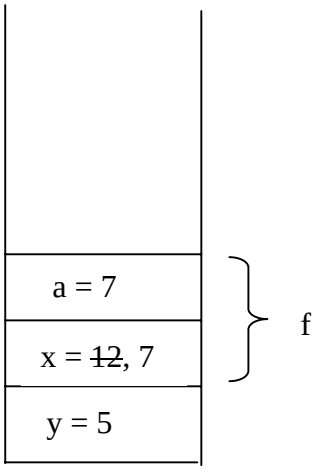


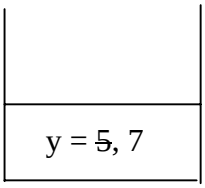
Risultato finale: y = 12

Passaggio per risultato (con valore iniziale di default per il parametro 5)









Risultato finale: y = 7