GUIDA ALLO SVOLGIMENTO DEGLI ESERCIZI SUL PASSAGGIO DI PARAMETRI

Gli esercizi sul passaggio di parametri presenti nelle tracce d'esame sono relativamente semplici da svolgere. Tuttavia occorre prestare la massima attenzione durante lo svolgimento poichè basta sbagliare un solo numero per falsare buona parte del risultato.

Iniziamo con una spiegazione teorica per poi passare allo svolgimento di un esercizio.
Nella traccia viene fornito un programma del quale occorre scrivere l'output e disegnare gli stack di attivazione per ogni caso richiesto
Gli elementi da valutare durante lo svolgimento dell'esercizio sono essenzialmente 2, cioè lo **SCOPING** ed il **MODE**

SCOPING

Il tipo di **SCOPING** indica l'ambiente **non locale** (cioè le variabili non dichiarate o ricevute come parametro all'interno della procedure) al quale ciascuna procedure del programma deve fare riferimento. Lo scoping può essere di 2 tipi:

- **SCOPING STATICO:** Per le variabili non locali, la procedure deve fare riferimento al blocco della procedure che la contiene sintatticamente. Per fare questo è sufficiente dare un'occhiata all'indentazione del codice per capire subito a quale blocco occorre fare riferimento.

- **SCOPING DINAMICO:** Per le variabili non locali, la procedure deve fare riferimento al blocco della procedure chiamante

MODE

Il tipo di **MODE** indica la modalità di passaggio dei parametri al momento della chiamata. I tipi di MODE possibili sono 6:

- **IN/OUT PER COPIA:** la variabile passata dal chiamante viene copiata in una variabile locale della procedura chiamata e ciascuna modifica apportata ha effetto esclusivamente sulla variabile locale della procedura chiamata. Quando la procedura chiamata termina, il valore della variabile viene restituito al chiamante e copiato nella relativa variabile locale del chiamante.

- **IN PER COPIA:** la variabile passata dal chiamante viene copiata in una variabile locale della procedura chiamata e ciascuna modifica apportata ha effetto esclusivamente sulla variabile locale della procedura chiamata.

- **IN/OUT PER RIFERIMENTO:** nella variabile della procedure chiamata viene copiato un riferimento alla variabile della procedure chiamante. La variabile del chiamante è accessibile sia in "lettura" che in "scrittura" e qualunque modifica apportata nella procedure chiamata ha immediatamente effetto sul valore della variabile della procedure chiamante.

- **IN PER RIFERIMENTO:** nella variabile della procedure chiamata viene copiato un riferimento alla variabile della procedure chiamante. La variabile del chiamante è accessibile solo in "lettura" e non può essere dunque modificata.
  ATTENZIONE: quando il MODE è di tipo *IN PER RIFERIMENTO*, non è possibile modificare il valore della variabile all'interno della procedure chiamata. L'istruzione della procedure chiamata che tenta di modificare una variabile passata dal chiamante con modalità *IN PER RIFERIMENTO*, causa un errore.

Per i MODE **OUT per copia** ed **OUT per riferimento** riporto quanto scritto nelle slide del docente:
Alcuni linguaggi assumono che i parametri OUT non siano inizializzati e ne proibiscono la "lettura", ad esempio:

- uso a destra di un assegnamento
- passaggio a un parametro *IN* o *IN/OUT* di un'altra procedura

Non esistono regole generali nemmeno tra diverse versioni di uno stesso linguaggio:

- Ada 83 proibisce di "leggere" i parametri OUT
- Le versioni successive invece lo permettono

NOTA: negli esercizi proposti dal docente, la lettura di parametri OUT non inizializzati genera errore.

ESERCIZIO

Passiamo adesso a mostrare un esercizio svolto.
L'esercizio fa parte della prova d'esame del 2 marzo 2017:

Esercizio sul passaggio di parametri – Max 11 punti

Dire qual è l'output del seguente programma nei casi elencati qui sotto:

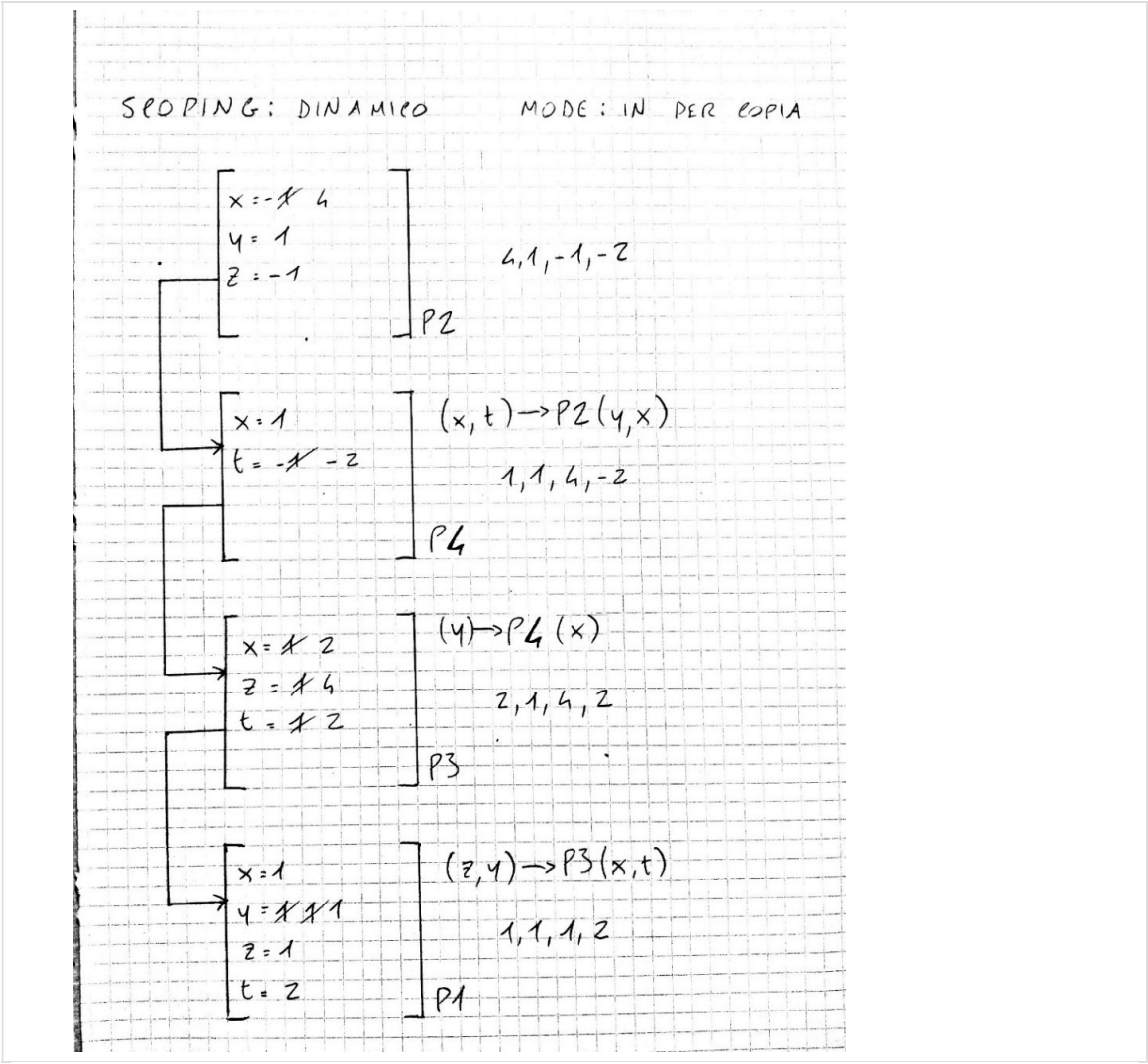
1. Scoping dinamico, [MODE] = IN per copia
2. Scoping dinamico, [MODE] = IN per riferimento
3. Scoping statico, [MODE] = IN per copia
4. Scoping statico, [MODE] = IN OUT per riferimento

Mostrare gli stack di attivazione (pena la perdita di punti), tranne nei casi di errore, nei quali bisogna invece indicare l'istruzione che causa l'errore.

```
program p1
int x; int y; int z; int t;
     procedure p2([IN x copia] int y,[IN x copia] int x)
     int z;
     BEGIN
     z=x;
     x=4;
     t=z-1;
     write(x,y,z,t);
     END

     procedure p3([MODE] int x,[IN x copia] int t)
     int z;
          procedure p4([IN x copia] int x)
          int t;
          BEGIN
          t=x-2;
          z=4;
          y=y;
          p2(x, t);
          write(x,y,z,t);
          END

     BEGIN
     z=t;
     x=z*2;
     t=x;
     y=z*1;
     p4(y);
     write(x,y,z,t);
     END

BEGIN
x=1;
y=1;
z=1;
t=2;
p3(z, y);
write(x,y,z,t);
END
```

L'esercizio richiede di indicare l'output del programma e disegnare gli stack di attivazione per 4 diverse combinazioni di **SCOPING**e **MODE**.
NOTA: negli svolgimenti seguenti sono già scritti tutti i valori assunti dalle variabili. Chiaramente, partendo dall'inizio dell'esercizio, il valore delle variabili viene aggiornato di volta in volta durante lo svolgimento.

PUNTO 1- SCOPING: dinamico **MODE:** IN per copia



Innanzitutto spieghiamo gli elementi utilizzati in questa rappresentazione grafica degli stack di attivazione.
Ciascuna procedure è rappresentata con un blocco formato da due parentesi quadre.
Le frecce alla sinistra di ogni blocco indicano lo SCOPING. Trattandosi in questo caso di scoping dinamico è ovvio che ogni freccia andrà verso il blocco immediatamente più in basso (il blocco del chiamante)
Le lettere nel blocco rappresentano le variabili ed i relativi valori da esse assunti durante l'esecuzione del programma.
Alla destra del blocco, dal basso verso l'alto, si trovano il nome della procedure, l'output della procedure e la corrispondenza delle variabili di procedure chiamante e chiamata (quest'ultima io la scrivo per avere più informazioni sotto mano ma volendo si può vedere diretamente dalla traccia mentre si svolge l'esercizio).
Andiamo ad analizzare i blocchi delle singole procedure:

- **PROCEDURE P1:** La procedure P1 è la prima ad essere eseguita. Le variabili x,y,z,t vengono inizializzate rispettivamente ai valori 1,1,1,2 e viene chiamata la procedure P3 passando i valori z,y di P1 alle variabili x,t di P3

- **PROCEDURE P3:** Il chiamante P1 passa a P3 i valori (1,1) che vengono copiati nelle variabili (x,t) con modalità IN per copia ed anche la variabile z viene dichiarata localmente all'interno del blocco. Per la variabile non locale y, invece, occorre fare riferimento allo scoping. Trattandosi in questo caso di scoping dinamico, va considerata la variabile y del chiamante, e quindi la y del blocco P1. All'inizio del blocco saranno quindi inizializzate soltanto le variabili x=1 e t=1.
  Vediamo ora come cambia il valore delle variabili per ogni istruzione:
  z=t ---> alla variabile locale z viene assegnato il valore della variabile t, quindi sarà z=1
  x=z*2 ---> analogamente il valore di x sarà x=1*2 cioè x=2
  t=x ---> t=2
  y=z*1 ---> y=1 come detto, y non è una variabile locale poichè non è stata dichiarata nel blocco P3, dunque si fa riferimento allo scoping e si modifica il valore della y del blocco P1 (che era già 1 ma va barrato e riscritto).
  A questo punto le variabili x,z,t di P3 hanno valore 2,1,2 e viene chiamata la procedure P4 passando il valore di y alla variabile x di P4

- **PROCEDURE P4:** Il chiamante P3 passa il valore 1 della variabile y alla variabile locale x di P4 con modalità IN per copia. Come detto la variabile y non è una variabile locale di P3 e quindi a P4 viene passato il valore della y di P1 a cui P3 fa riferimento per lo scoping. Localmente viene dichiarata anche la variabile t, mentre per le variabili y e z si dovrà fare riferimento alle variabili omonime di P3 in virtù dello scoping dinamico.
  Analizziamo le istruzioni:
  t=x-2 ---> t=-1
  z=4 ---> non essendo una variabile locale, va modificata la z di P3 e posta uguale a 4
  y=y ---> la y di P4 non è una variabile locale e si fa riferimento alla y di P3 che a sua volta fa riferimento alla y di P1. Dunque questa assegnazione assegna ancora una volta il valore 1 alla variabile y di P1.
  A questo punto viene effettuata una chiamata a P2 passando i valori di (x,t) alle variabili (y,x) di P2

- **PROCEDURE P2:** La procedure P2 ha come variabili locali x,y,z. I valori di y e x sono passati dal chiamante e sono y=1 e x=-1. Analizziamo le istruzioni:
  z=x ---> z=-1
  x=4
  t=z-1 ---> t non è una variabile locale e dunque va modificata la variabile t del chiamante P4, quindi t=-1-1 cioè t=-2. Da notare che la variabile z utilizzata in questa istruzione è comunque la variabile locale z di P2. Sottolineamo che lo scoping riguarda solo ed esclusivamente le variabili non locali.

A questo punto, andando a ritroso tutte le procedure eseguono la **write(x,y,z,t)** stampando l'output che, guardando l'ultimo valore assunto da ciascuna variabile, sarà il seguente:

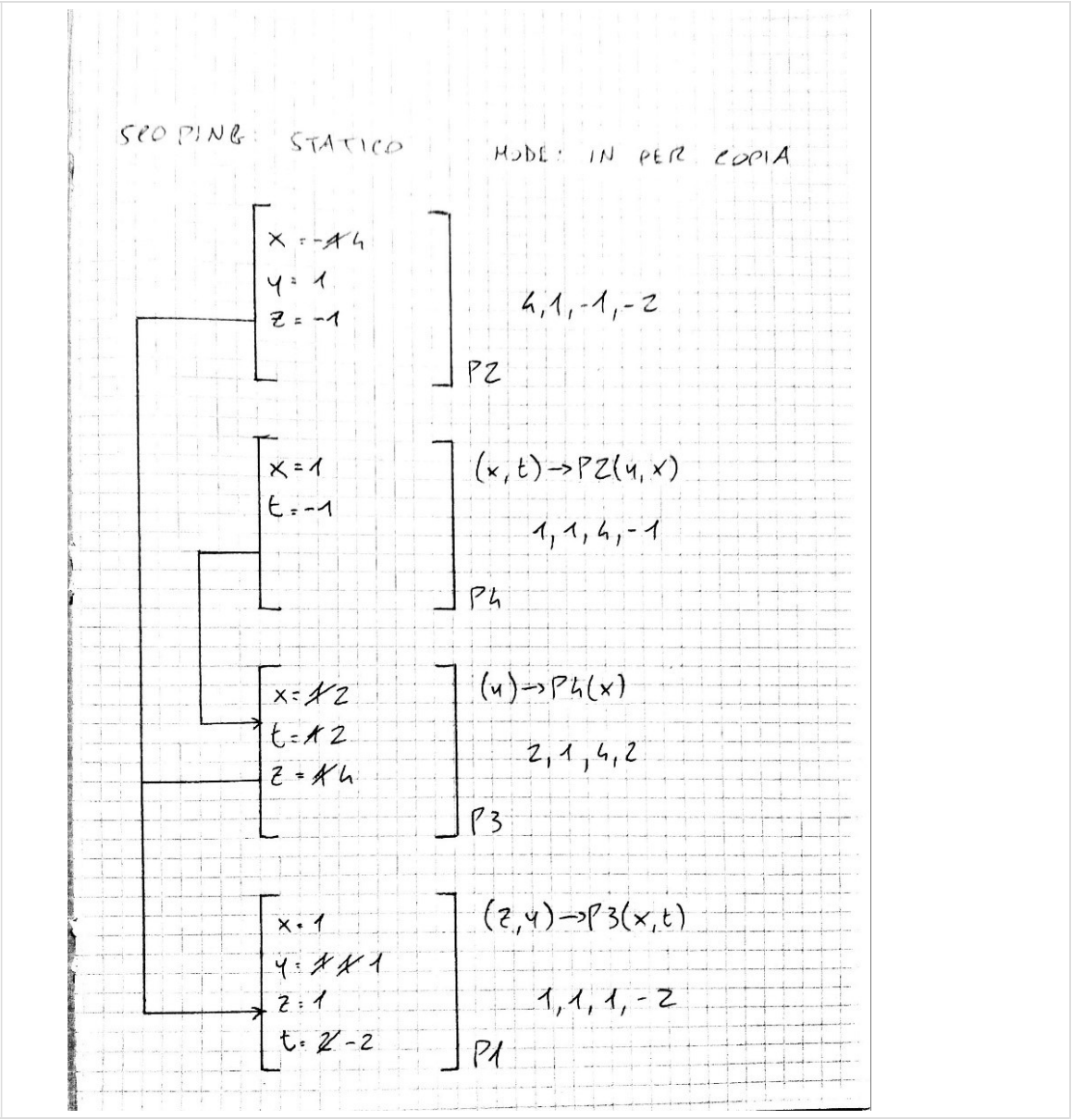
P2= 4,1,-1,2
P4=1,1,4,-2
P3=2,1,4,2
P1=1,1,1,2

NOTA: anche per la write(x,y,z,t) si fa riferimento allo scoping per le variabili non locali.

PUNTO 2- SCOPING: dinamico **MODE:** IN per riferimento
Il **99,9%** delle volte, negli esercizi sul passaggio di parametri, il programma della traccia è scritto in modo da dare errore nel caso di passaggio con modalità IN per riferimento. Come detto all'inizio,un'istruzione che tenta di modificare una variabile passata con modalità ***IN PER RIFERIMENTO***, causa un errore. In questi casi quindi, basta controllare se c'è una istruzione che tenta di modificare una variabile ricevuta con modalità IN per riferimento. Nell'esercizio dell'esempio, occorre guardare la procedure P3 e più precisamente la sua variabile **x**. Se troviamo un'istruzione che tenta di modificare il valore di tale variabile, allora possiamo dire che

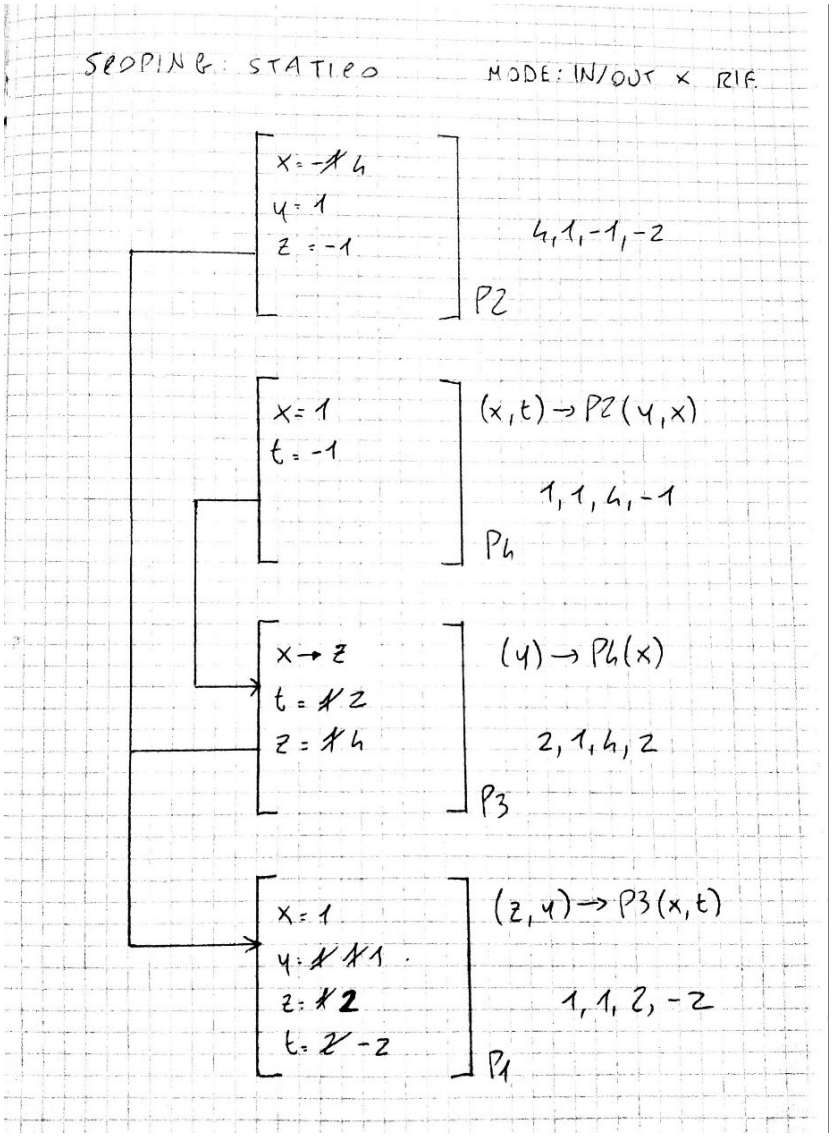
quella istruzione causa errore. Nel nostro caso quindi **l'istruzione x=z*2 di P3 causa errore e NON vanno indicati output e stack di attivazione.**

PUNTO 3- SCOPING: statico **MODE:** IN per copia



Lo svolgimento di questo punto è del tutto simile a quello del punto 1. L'unica differenza sta nel tipo di SCOPING. In questo caso lo scoping è di tipo statico e dunque per le variabili non locali ogni procedure deve fare riferimento al blocco che la contiene sintatticamente. Dando uno sguardo all'indentazione del codice si capisce subito che P2 e P3 sono contenute in P1 mentre P4 è contenuta in P3. Quindi, durante lo svolgimento, aggiungiamo le frecce alla sinistra dei blocchi per poter vedere subito lo scoping di ciascun blocco senza dover ogni volta ricontrollare la traccia. In questo caso, per esempio, l'assegnamento **t=z-1** nel blocco P2 va a modificare la variabile t del blocco P1.

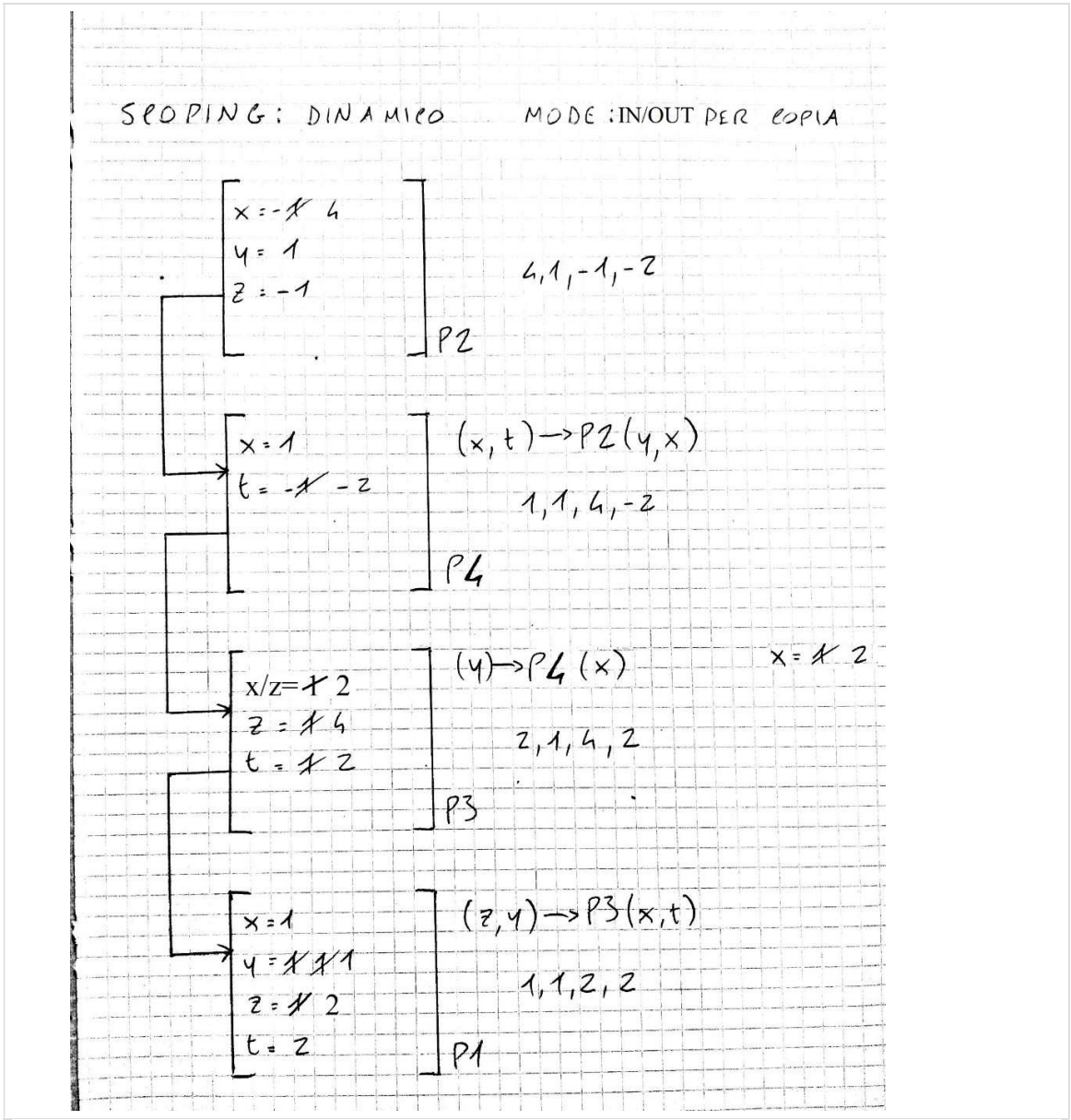
PUNTO 4- SCOPING: statico **MODE:** IN/OUT per riferimento



Nel blocco P3 ho utilizzato la scrittura **x-->z** per indicare che la variabile locale x contiene un riferimento alla variabile z del chiamante (cioè P1).
Anche in questo caso lo svolgimento è analogo a quello dei punti precedenti. L'unica differenza sta nella modalità di passaggio della variabile x di P3 che è di tipo IN/OUT per riferimento. In questo caso qualsiasi modifica della variabile x di P3 comporta l'immediata modifica della variabile z di P1 a cui fa riferimento. Per esempio, l'assegnamento **x=z*2** in P3 comporta l'assegnamento del valore z*2 alla variabile **z** di P1
ATTENZIONE: occorre stare attenti a non confondere lo scoping delle variabili non locali con il riferimento delle variabili passate con modalità IN/OUT per riferimento

Poiche nell'esercizio precedente non ci sono le modalità di passaggio **IN/OUT per copia**, **OUT per copia** ed **OUT per riferimento**, ne riporto un esempio a titolo esplicativo.

Supponiamo che il punto 1 dell'esercizio fosse stato:
SCOPING: dinamico e MODE: IN/OUT per copia
Lo svolgimento dell'esercizio sarebbe stato il seguente:



La scrittura **x/z** nel blocco P3 indica che al termine della procedure P3 il valore della variabile x deve essere copiato nella variabile z del chiamante, in questo caso P1.
Quindi durante lo svolgimento dell'esercizio, dopo aver scritto l'output del blocco P3 assegneremo alla variabile z di P1 il valore della variabile x di P3, quindi z=2

Vediamo due casi di passaggio OUT per riferimento:

- **CASO CON ERRORE (PUNTO 4):**

Dire qual è l'output del seguente programma nei casi elencati qui sotto:

1. Scoping dinamico, [MODE] = IN per copia
2. Scoping statico, [MODE] = IN per copia
3. Scoping statico, [MODE] = IN OUT per riferimento
4. Scoping statico, [MODE] = OUT per riferimento

Mostrare gli stack di attivazione (pena la perdita di punti), tranne nei casi di errore, nei quali bisogna invece indicare l'istruzione che causa l'errore.

```
program p1
int a; int b; int c; int d;
     procedure p2([IN OUT x copia] int a)
     int c; int d;
     BEGIN
     if b>3 then c=b else c=a+2;
     d=4;
     a=c-4;
     if a>10 then b=c else b=d-3;
     write(a,b,c,d);
     END

     procedure p3([MODE] int b,[IN x copia] int c)
     int d;
          procedure p4([IN x copia] int d)
          int a; int c;
          BEGIN
          a=b*4;
          c=3;
          d=b-4;
          b=d;
          p2(b);
          write(a,b,c,d);
          END

     BEGIN
     d=a;
     b=b+2;
     c=a*3;
     a=1;
     p4(b);
     write(a,b,c,d);
     END

BEGIN
a=0;
b=3;
c=0;
d=2;
p3(d, c);
write(a,b,c,d);
END
```

In questo caso, l'assegnamento **b=b+2** nel blocco P3 è **illegale** perchè si tenta di leggere il valore della variabile **b** che non è inizializzata.

- **CASO SENZA ERRORE (PUNTO 2):**

Dire qual è l'output del seguente programma nei casi elencati qui sotto:

1. Scoping dinamico, [MODE] = IN per copia
2. Scoping dinamico, [MODE] = OUT per riferimento
3. Scoping statico, [MODE] = IN per copia
4. Scoping statico, [MODE] = IN OUT per copia

Mostrare gli stack di attivazione (pena la perdita di punti), tranne nei casi di errore, nei quali bisogna invece indicare l'istruzione che causa l'errore.

```
program p1
int q; int r; int s; int t;
     procedure p2([IN x copia] int s,[IN x copia] int q)
     int r;
          procedure p3([IN OUT x rif] int r)
          int q;
          BEGIN
          q=r*3;
          r=r+2;
          s=2;
          t=q;
          p4(t, s);
          write(q,r,s,t);
          END

     BEGIN
     if s<5 then r=q*2 else r=4;
     s=r;
     q=1;
     t=3;
     p3(t);
     write(q,r,s,t);
     END

     procedure p4([MODE] int q,[IN x copia] int t)
     int s;
     BEGIN
     s=4;
     q=2;
     t=s+4;
     r=t;
     write(q,r,s,t);
     END

BEGIN
q=4;
r=3;
s=0;
t=1;
p2(r, q);
write(q,r,s,t);
END
```

In questo caso non c'è errore poichè non c'è nessuna lettura del valore della variabile q non inizializzata. Lo svolgimento, in questo caso, è del tutto uguale alla modalità IN/OUT per riferimento.

Ho controllato tutte le prove presenti sul sito del docente ma non ho trovato alcuna traccia contenente la modalità OUT per copia, quindi mi limito a ribadire quanto detto in precedenza, cioè che l'eventuale lettura di variabili non inizializzate genera errore.

Per completare la guida riporto questa tabella fatta da **luigi_starace** che specifica quali errori possono verificarsi per ogni modalità di passaggio dei parametri:

MODALITÀ	Errore in compilazione
IN PER COPIA	MAI
IN PER RIFER.	Se c'è una scrittura sul parametro
OUT PER COPIA	Se c'è una lettura del valore del parametro
OUT PER RIFER.	Se c'è una lettura del valore del parametro
IN/OUT PER COPIA	MAI
IN/OUT PER RIFER.	MAI