Esercizio 1

Data la seguente tabella di operatori aritmetici,

Operatori	Associatività
**	destra
*, /	destra
+, -	sinistra

riscrivere la seguente espressione in forma parentetica:

```
a + b – c * d + e / f * g ** h ** i
```

Esercizio 1

Data la seguente tabella di operatori aritmetici,

Operatori	Associatività
**	destra
*, /	destra
+, -	sinistra

riscrivere la seguente espressione in forma parentetica:

```
a + b – c * d + e / f * g ** h ** i
```

```
(((a + b) – (c * d)) + (e / (f * (g ** (h ** i)))))
```

Esercizio 2

Data la seguente tabella di descrizione di operatori aritmetici,

Operatori	Associatività
^	destra
*, /	sinistra
+, -	destra

riscrivere la seguente espressione in forma parentetica:

```
a + b – c * d + e ^ f ^ g + h / i / l + m * n
```

Esercizio 2

Data la seguente tabella di descrizione di operatori aritmetici,

Operatori	Associatività
^	destra
*, /	sinistra
+, -	destra

riscrivere la seguente espressione in forma parentetica:

```
a + b – c * d + e ^ f ^ g + h / i / l + m * n
```

```
(a + (b – ((c * d) + ((e ^ (f ^ g)) + (((h / i) / l) + (m * n))))))
```

Esercizio 3

Specificare la semantica operazionale del seguente assegnamento, supponendo di valutare in corto circuito l'espressione di destra:

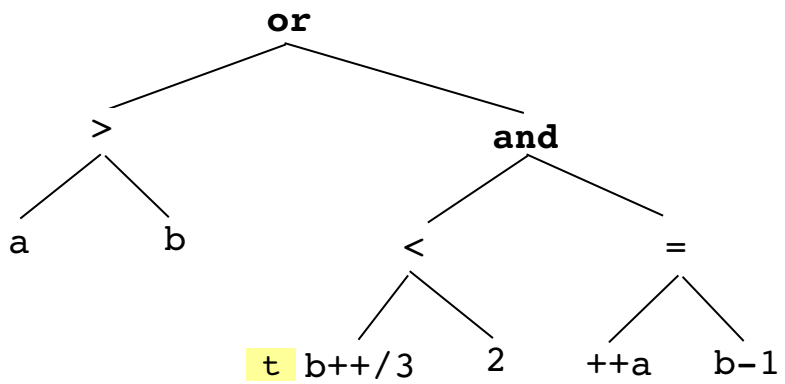
```
x := (a > b) or ((b++ / 3) < 2 and ++a = b-1);
```

Esercizio 3

Specificare la semantica operazionale del seguente assegnamento, supponendo di valutare in corto circuito l'espressione di destra:

```
x := (a > b) or ((b++ / 3) < 2 and ++a = b-1);
```

```
if a > b then x := true
else
{
    t := b/3;
    b := b + 1;
    if t >= 2 then x := false
    else
    {
        a := a+1;
        if a != b-1 then x := false
        else
            x := true
    }
}
```



Esercizio 4

Assumendo la seguente tabella di operatori (in cui ogni operatore valuta i suoi operandi da sinistra a destra e la valutazione delle espressioni logiche è in corto circuito),

Operatori	Associatività
++, --	nonassoc
*, /	sinistra
+, -	sinistra
<, <=, >, >=	nonassoc
and, **or**	sinistra
?:	nonassoc

specificare la semantica operazionale del seguente assegnamento:

```
x = (++a+b<=c and a--/3+c>++d ? a++ : ++b/2)
```

NB: Il linguaggio di specifica non contempla gli operatori ++, --.

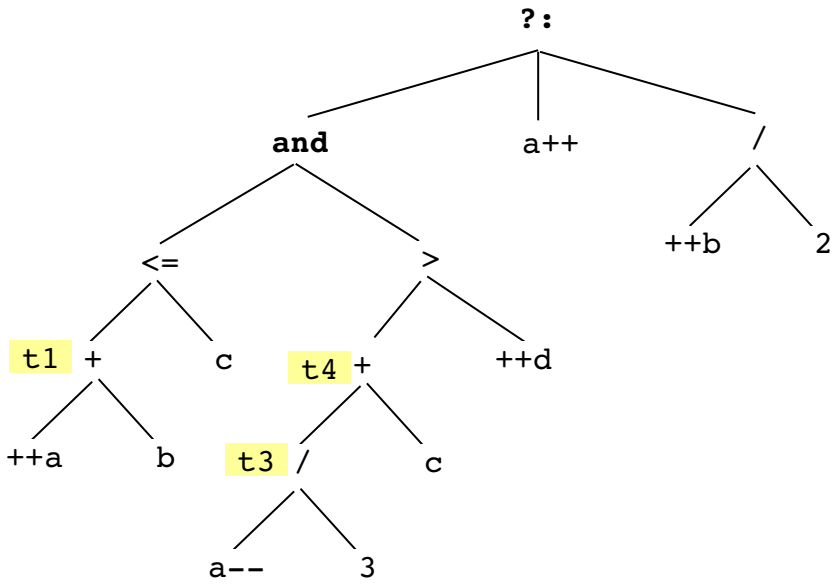
Operatori	Associatività
++, --	nonassoc
*, /	sinistra
+, -	sinistra
<, <=, >, >=	nonassoc
and, **or**	sinistra
?:	nonassoc

Esercizio 4

```
x = (++a+b<=c and a--/3+c>++d ? a++ : ++b/2)
```

```
x = (((((++a)+b)<=c) and ((((a--)/3)+c)>(++d))) ? (a++) : ((++b)/2))
```

```
a := a+1;
t1 := a+b;
if t1 > c then
{   b := b+1; x := b/2;
}
else
{   t2 := a;
    a := a-1;
    t3 := t2/3;
    t4 := t3+c;
    d := d+1;
    if t4 > d then
    {  x := a; a := a+1;
    }
    else
    {  b := b+1; x := b/2;
    }
};
```



Esercizio 5

Assumendo la seguente tabella di operatori (in cui ogni operatore binario valuta prima l'operando di destra e poi quello di sinistra, e la valutazione delle espressioni logiche è in corto circuito),

Operatori	*Associatività*
^	destra
+, -, *, /	sinistra
<, >	nonassoc
and, **or**	sinistra

specificare la semantica operazionale del seguente assegnamento della variabile logica v:

```
v := a + b * c > d ^ e ^ f + 1 or g < h/i
```

NB: Il linguaggio di specifica operazionale contempla le seguenti limitazioni:

- Non contiene gli operatori logici **and**, **or**;
- Contiene gli operatori ^, +, -, *, /, <, >, che però non possono essere applicati al risultato di altre operazioni (quindi è necessario l'introduzione di opportuni temporanei per i risultati intermedi …).

Esercizio 5

Assumendo la seguente tabella di operatori (in cui ogni operatore binario valuta prima l'operando di destra e poi quello di sinistra, e la valutazione delle espressioni logiche è in corto circuito),

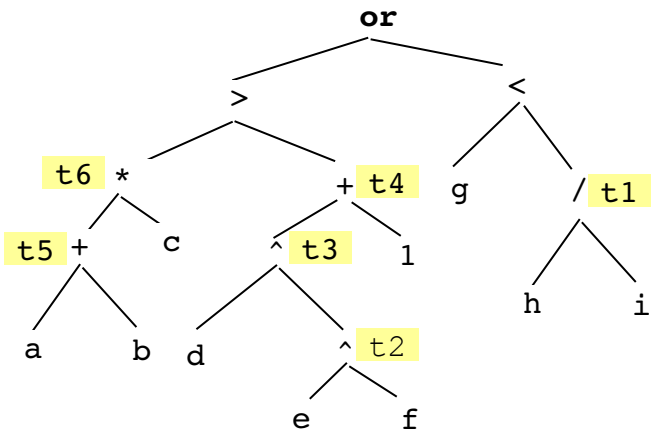
Operatori	*Associatività*
^	destra
+, -, *, /	sinistra
<, >	nonassoc
and, **or**	sinistra

specificare la semantica operazionale del seguente assegnamento della variabile logica v:

```
v := a + b * c > d ^ e ^ f + 1 or g < h/i
```

```
v := (((a + b) * c) > ((d ^ (e ^ f)) + 1)) or (g < (h/i))
```

```
t1 = h/i;
if g < t1 then
  v := true
else
{   t2 = e^f;
    t3 = d^t2;
    t4 = t3+1;
    t5 = a+b;
    t6 = t5 * c;
    if t6 > t4 then v := true else v := false
}
```



Esercizio 6

Assumendo la seguente tabella di operatori (in cui ogni operatore binario valuta prima l'operando di sinistra e poi quello di destra, e la valutazione delle espressioni logiche è in corto circuito),

Operatori	*Associatività*
*, /	destra
+, -	sinistra
<, >	nonassoc
and, **or**	sinistra

specificare la semantica operazionale del seguente assegnamento (mediante espressione condizionale) della variabile logica x, in cui l'espressione condizionale è valutata da sinistra a destra:

```
x := (a * b / c + 3 * d > e + 1 / f or y ? z : p() and q())
```

NB: Il linguaggio di specifica operazionale contempla le seguenti limitazioni:

- Non contiene gli operatori logici;
- Contiene gli operatori +, -, *, /, <, >, che però possono essere applicati solo a costanti o variabili.

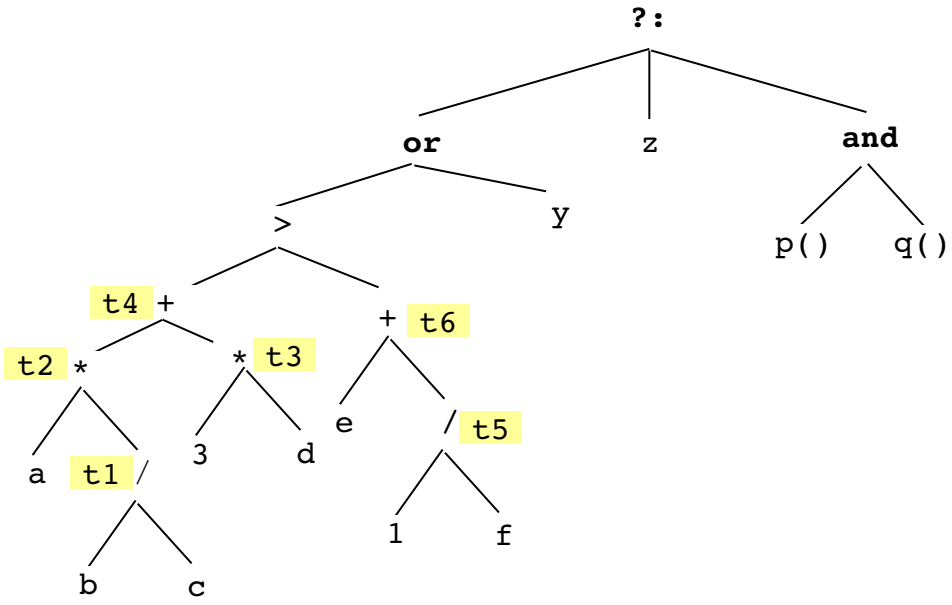
Esercizio 6

Operatori	*Associatività*
*, /	destra
+, -	sinistra
<, >	nonassoc
and, **or**	sinistra

```
x := (a * b / c + 3 * d > e + 1 / f or y ? z : p() and q())
```

```
x := (((((a * (b / c)) + (3 * d)) > (e + (1 / f))) or y) ? z : (p() and q()))
```

```
t1 := b / c;
t2 := a * t1;
t3 := 3 * d;
t4 := t2 + t3;
t5 := 1 / f;
t6 := e + t5;
if t4 > t6 then x := z
else
{ if y then x := z
  else
  { t7 := p(); if t7 then x := q() else x := false }
}
```



Esercizio 7

Assumendo la seguente tabella di operatori (in cui la priorità degli operatori decresce dall'alto verso il basso, ogni operatore binario valuta prima l'operando di sinistra e poi quello di destra, e la valutazione delle espressioni logiche è in corto circuito),

Nome	*Operatore*	*Tipo*	*Associatività*
selezione, proiezione	σ, π	unario	destra
intersezione	$\cap$	binario	sinistra
unione, differenza	$\cup$, $-$	binario	sinistra
inclusione	$\supset$, $\subset$	binario	nonassoc
congiunzione, disgiunzione	$\wedge$, $\vee$	binario	sinistra

specificare la semantica operazionale del seguente assegnamento della variabile complessa `R`:

$$R := \sigma\ \pi\ \sigma\ A \cup \sigma\ B - C \supset D \cup E \cap F \vee G \subset \pi\ \sigma\ H$$

NB: Il linguaggio di specifica operazionale contempla le seguenti limitazioni:

- Non contiene gli operatori logici $\wedge$, $\vee$;
- Contiene tutti gli operatori insiemistici, che però possono essere applicati solo a variabili.

Esercizio 7

Assumendo la seguente tabella di operatori (in cui la priorità degli operatori decresce dall'alto verso il basso, ogni operatore binario valuta prima l'operando di sinistra e poi quello di destra, e la valutazione delle espressioni logiche è in corto circuito),

Nome	*Operatore*	*Tipo*	*Associatività*
selezione, proiezione	σ, π	unario	destra
intersezione	∩	binario	sinistra
unione, differenza	∪, −	binario	sinistra
inclusione	⊃, ⊂	binario	nonassoc
congiunzione, disgiunzione	∧, ∨	binario	sinistra

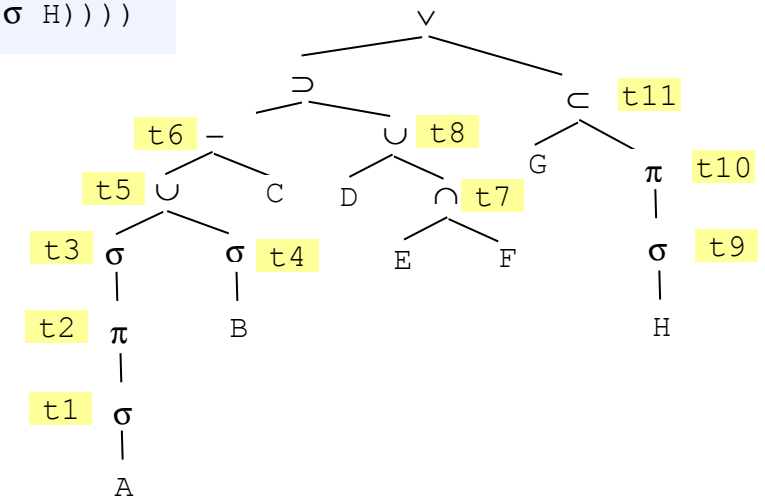
specificare la semantica operazionale del seguente assegnamento della variabile complessa `R`:

```
R := σ π σ A ∪ σ B − C ⊃ D ∪ E ∩ F ∨ G ⊂ π σ H
```

```
R := (((((σ(π(σ A))) ∪ (σ B)) − C) ⊃ (D ∪ (E ∩ F))) ∨ (G ⊂ (π(σ H))))
```

```
t1 := σ A;
t2 := π t1;
t3 := σ t2;
t4 := σ B;
t5 := t3 ∪ t4;
t6 := t5 − C;
t7 := E ∩ F;
t8 := D ∪ t7;
```

```
if t6 ⊃ t8 then R := true
else
{
  t9 := σ H;
  t10 := π t9;
  t11 := G ⊂ t10;
  R := t11;
}
```



Esercizio 8

È data la seguente tabella di operatori (con priorità decrescente dall'alto verso il basso e valutazione delle espressioni logiche in corto circuito):

Operatori	*Associatività*	*Ordine valutazione operandi*
+, **–**	sinistra	da sinistra a destra
*****	destra	da sinistra a destra
<, **>**	nonassoc	da destra a sinistra
and, **or**	sinistra	da destra a sinistra

e la seguente istruzione di assegnamento:

```
x := a + b * c – d * e > f + (g * h) – i * m and v > w or c < d
```

a) Rappresentare l'albero della espressione di assegnamento.
b) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale contempla le seguenti limitazioni:

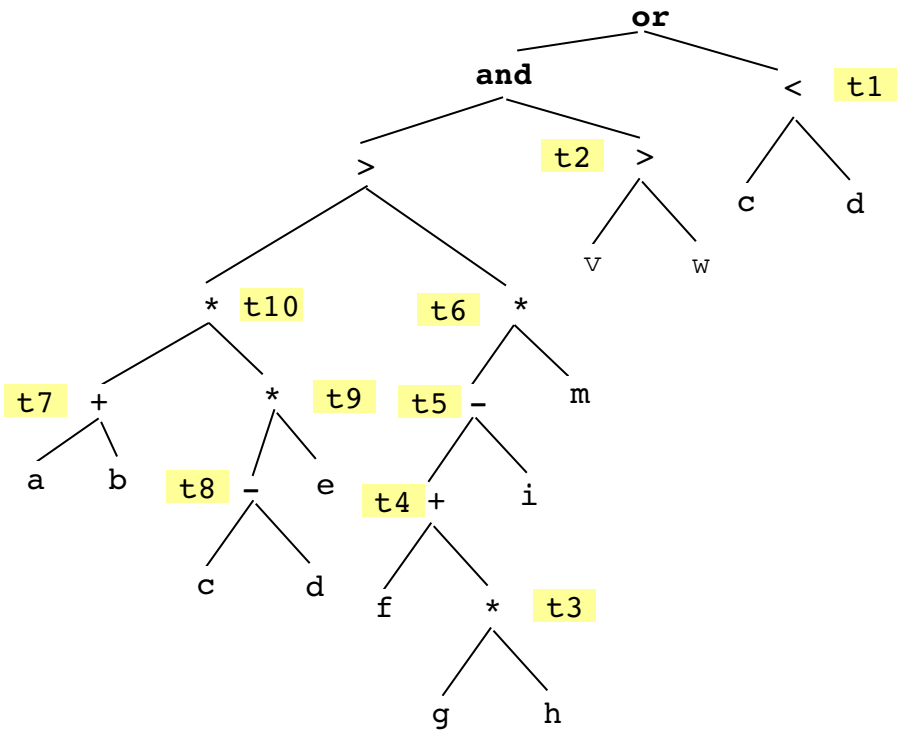
- Non contiene gli operatori logici **and**, **or**;
- Contiene gli operatori **+**, **-**, *****, **<**, **>** che però non possono essere applicati al risultato di altre operazioni (necessità di variabili temporanee);
- Contiene le istruzioni condizionali **if** *cond* **then** … **endif** e **if** *cond* **then** … **else** … **endif**, in cui *cond* può essere o un temporaneo (ad esempio, **if** t1 **then** …) oppure la negazione di un temporaneo (ad esempio, **if** !t1 **then** …).

Esercizio 8

Operatori	*Associatività*	*Ordine valutazione operandi*
+, –	sinistra	da sinistra a destra
*****	destra	da sinistra a destra
<, >	nonassoc	da destra a sinistra
and, **or**	sinistra	da destra a sinistra

```
x := a + b * c – d * e > f + (g * h) – i * m and v > w or c < d
```

```
x := (((((a + b) * ((c – d) * e)) > (((f + (g * h)) – i) * m)) and (v > w)) or (c < d))
```




```
t1 := c < d;
if t1 then
  x := true
else
  t2 := v > w;
  if !t2 then
    x := false
  else
    t3 := g * h;
    t4 := f + t3;
    t5 := t4 – i;
    t6 := t5 * m;
    t7 := a + b;
    t8 := c – d;
    t9 := t8 * e;
    t10 := t7 * t9;
    x := t10 > t6
  endif
endif.
```

Esercizio 9

È data la seguente tabella di operatori per la quale si assume priorità decrescente dall'alto verso il basso, valutazione degli operandi da sinistra a destra e valutazione delle espressioni logiche in corto circuito:

Operatori	*Associatività*
`*`	sinistra
`+`	sinistra
`==`	nonassoc
`and`	sinistra
`or`	sinistra

e la seguente istruzione di assegnamento:

```
x := a + b + c * d * f == g * h + i or j == k and m + n == p
```

a) Rappresentare l'albero della espressione di assegnamento.
b) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

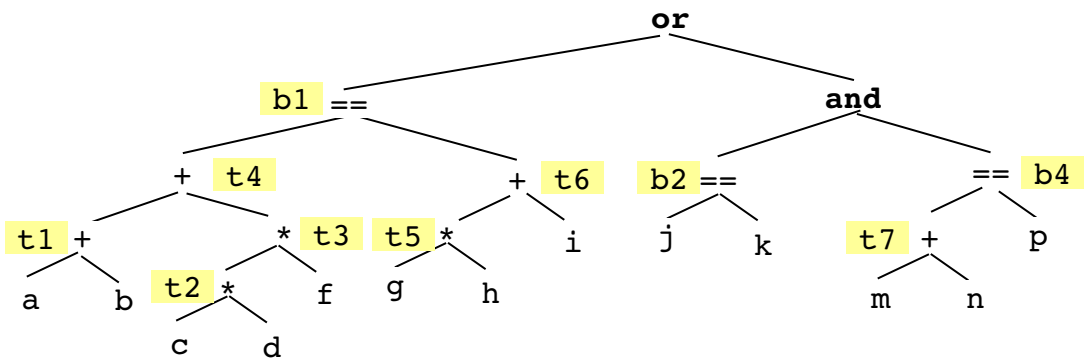
- Contiene gli operatori **+**, *****, **==**;
- Non contiene gli operatori logici **and**, **or**;
- Contiene l'operatore logico **not**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati sono variabili.

Esercizio 9

Operatori	*Associatività*
`*`	sinistra
`+`	sinistra
`==`	nonassoc
`and`	sinistra
`or`	sinistra

```
x := a + b + c * d * f == g * h + i or j == k and m + n == p
```

```
x := ((((a + b) + ((c * d) * f)) == ((g * h) + i)) or ((j == k) and ((m + n) == p)))
```




```
t1 := a + b;
t2 := c * d;
t3 := t2 * f;
t4 := t1 + t3;
t5 := g * h;
t6 := t5 + i;
b1 := t4 == t6;
if b1 then
  x := true
else
  b2 := j == k;
  b3 := not b2
  if b3 then
    x := false
  else
    t7 := m + n;
    b4 := t7 == p;
    x := b4
  endif
endif.
```

Esercizio 10

È data la seguente tabella di operatori per la quale si assume priorità decrescente dall'alto verso il basso:

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
`not`	unario	destra	-	-
`and`	binario	sinistra	da sinistra a destra	si
`or`	binario	sinistra	da destra a sinistra	no

e la seguente istruzione di assegnamento:

```
x := a and not b or c and d and e
```

a) Rappresentare l'albero della espressione di assegnamento.
b) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

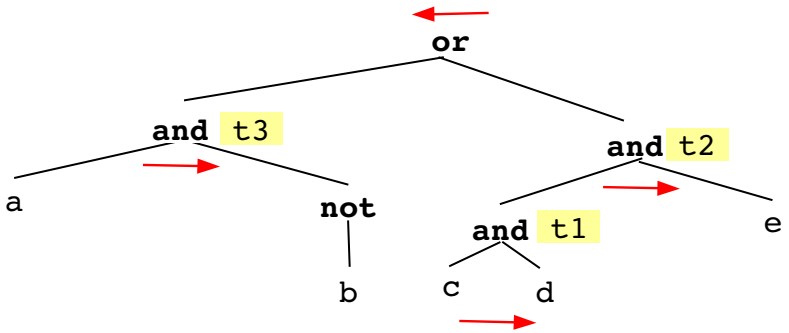
- Contiene gli operatori di negazione (**!**), disgiunzione (**||**) ed assegnamento (←).
- Non contiene l'operatore di congiunzione, né le costanti logiche **true**, **false**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati possono essere solo variabili.

Esercizio 10

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
not	unario	destra	-	-
and	binario	sinistra	da sinistra a destra	si
or	binario	sinistra	da destra a sinistra	no

```
x := a and not b or c and d and e
```

```
x := ((a and (not b)) or ((c and d) and e))
```




```
if c then t1 ← d else t1 ← c;
if t1 then t2 ← e else t2 ← t1;
if a then t3 ← !b else t3 ← a;
x ← t2 || t3.
```

Esercizio 11

È data la seguente tabella di operatori per la quale si assume priorità decrescente dall'alto verso il basso:

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
^	binario	destra	da destra a sinistra	si
*****	binario	sinistra	da sinistra a destra	si
+	binario	sinistra	da sinistra a destra	no
-	binario	sinistra	da sinistra a destra	no

Sono stabilite le seguenti regole di corto-circuito:

- $expr_1$ **^** $expr_2 = 1$ quando $expr_2 = 0$.
- $expr_1$ ***** $expr_2 = 0$ quando $expr_1 = 0$.

Quindi, data la seguente istruzione di assegnamento,

```
x := a + b – c + d – e * f * g ^ (h * i) ^ m
```

c) Rappresentare l'albero della espressione di assegnamento.
d) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

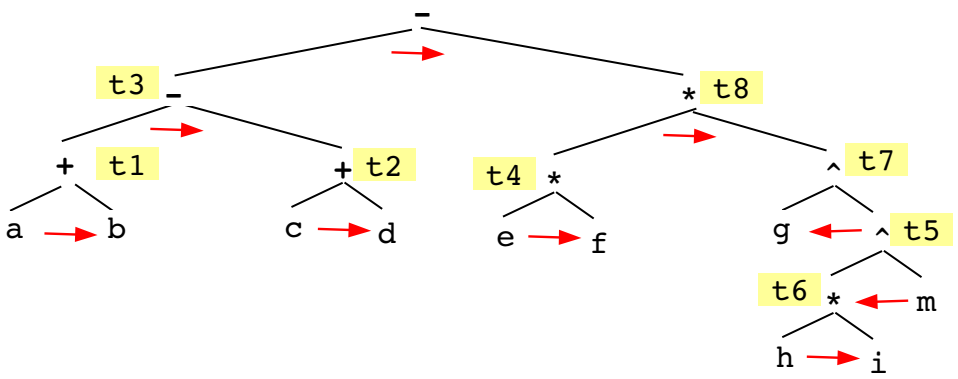
- Contiene gli operatori aritmetici **^**,*****,**+**,**-**.
- Contiene gli operatori di assegnamento '**=**' e di confronto di uguaglianza '**==**'.
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili o costanti.
- Contiene le istruzioni condizionali (*if-then-endif* ed *if-then-else-endif*) i cui predicati possono essere solo semplici confronti di uguaglianza.

Esercizio 11

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
^	binario	destra	da destra a sinistra	si
*	binario	sinistra	da sinistra a destra	si
+	binario	sinistra	da sinistra a destra	no
-	binario	sinistra	da sinistra a destra	no

```
x := a + b – c + d – e * f * g ^ (h * i) ^ m
```

```
x := (((a + b) – (c + d)) – ((e * f) * (g ^ ((h * i) ^ m))))
```




```
t1 = a + b;
t2 = c + d;
t3 = t1 - t2;
if e == 0 then t4 = 0 else t4 = e * f endif;
if t4 == 0 then
  t8 = 0
else
  if m == 0 then
    t5 = 1
  else
    if h == 0 then t6 = 0 else t6 = h * i endif;
    t5 = t6 ^ m
  endif;
  if t5 == 0 then t7 = 1 else t7 = g ^ t5 endif;
  t8 = t4 * t7
endif;
x = t3 – t8.
```

Esercizio 12

È data la seguente tabella di operatori per la quale si assume priorità decrescente dall'alto verso il basso:

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
`not`	unario	destra	-	-
`and`	binario	sinistra	da destra a sinistra	no
`or`	binario	destra	da sinistra a destra	si

e la seguente espressione:

```
a or b and c or d or not e
```

e) Rappresentare l'albero della espressione.
f) Specificare la semantica operazionale della espressione.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

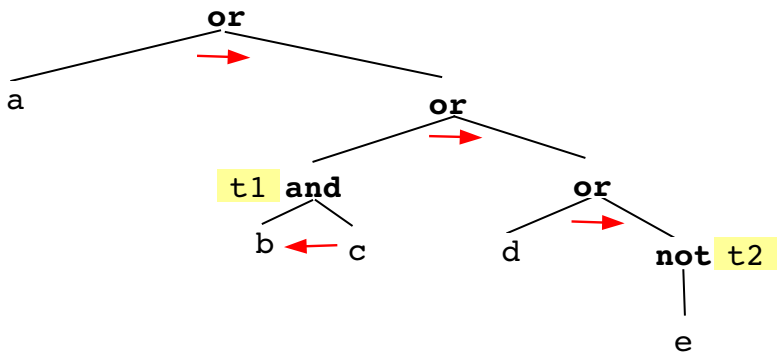
- Contiene gli operatori di negazione (**!**), congiunzione (**&&**) ed assegnamento (**:=**).
- Non contiene l'operatore di disgiunzione, né le costanti logiche **true**, **false**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati possono essere solo variabili;
- Contiene l'istruzione **return** il cui argomento è la variabile che contiene il valore della espressione;
- L'esecuzione della **return** termina immediatamente l'esecuzione del programma di specifica operazionale.

Esercizio 12

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
not	unario	destra	-	-
and	binario	sinistra	da destra a sinistra	no
or	binario	destra	da sinistra a destra	si

```
a or b and c or d or not e
```

```
(a or ((b and c) or (d or (not e))))
```




```
if a then return a;
t1 := c && b;
if t1 then return t1;
if d then return d;
t2 := !e;
return t2.
```

Esercizio 13

È data la seguente tabella di operatori, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
=	binario	-	da sinistra a destra	no
not	unario	destra	-	-
and	binario	sinistra	da sinistra a destra	si
or	binario	sinistra	da destra a sinistra	no
?:	ternario	-	da sinistra a destra	si

e la seguente espressione:

```
not a = b and c or d = e ? x and y : z
```

a) Rappresentare l'albero della espressione.
b) Specificare la semantica operazionale della espressione.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

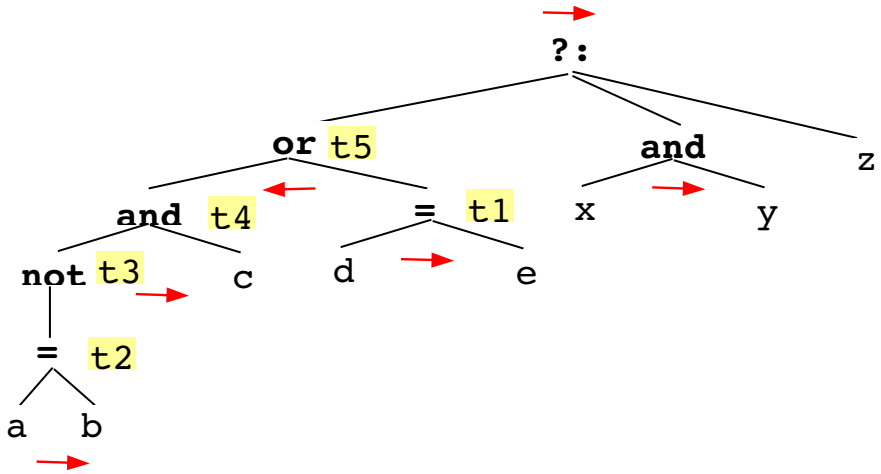
- Contiene gli operatori di negazione (**!**), disgiunzione (**||**), uguaglianza (**==**) ed assegnamento (**:=**);
- Non contiene l'operatore di congiunzione, ne l'operatore **?:**, ne le costanti logiche **true**, **false**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati possono essere solo variabili;
- Contiene l'istruzione **return** il cui argomento è la variabile che contiene il valore della espressione;
- L'esecuzione della **return** termina immediatamente l'esecuzione del programma di specifica operazionale.

Esercizio 13

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
=	binario	-	da sinistra a destra	no
not	unario	destra	-	-
and	binario	sinistra	da sinistra a destra	si
or	binario	sinistra	da destra a sinistra	no
?:	ternario	-	da sinistra a destra	si

```
not a = b and c or d = e ? x and y : z
```

```
((((not (a = b)) and c) or (d = e)) ? (x and y) : z)
```




```
t1 := d == e;
t2 := a == b;
t3 := !t2;
if t3 then t4 := c else t4 := t3 endif;
t5 := t4 || t1;
if t5 then
    if x then return y else return x endif
else
    return z
endif.
```

Esercizio 14

È data la seguente tabella di operatori, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
=	binario	-	da sinistra a destra	no
not	unario	destra	-	-
and, **or**	binario	sinistra	da sinistra a destra	si
?:	ternario	-	da destra a sinistra	no

e la seguente espressione:

```
(not x) = y or z = w and not p ? a or b : not c
```

a) Rappresentare l'albero della espressione.
b) Specificare la semantica operazionale della espressione.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

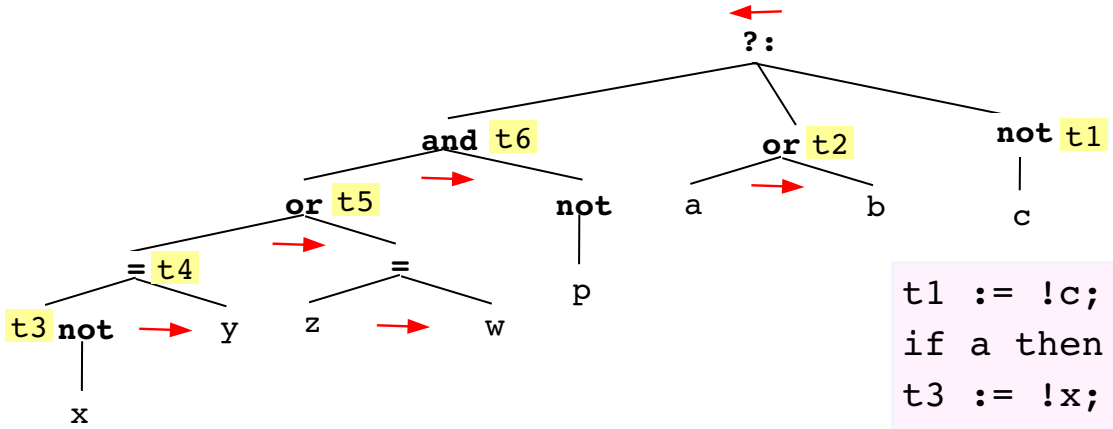
- Contiene gli operatori di negazione (**!**), uguaglianza (**==**) ed assegnamento (**:=**);
- Non contiene gli operatori **and**, **or**, ne l'operatore **?:**, ne le costanti logiche **true**, **false**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati possono essere solo variabili;
- Contiene l'istruzione **return** il cui argomento è la variabile che contiene il valore della espressione;
- L'esecuzione della **return** termina immediatamente l'esecuzione del programma di specifica operazionale.

Esercizio 14

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
=	binario	-	da sinistra a destra	no
not	unario	destra	-	-
and, **or**	binario	sinistra	da sinistra a destra	si
?:	ternario	-	da destra a sinistra	no

```
(not x) = y or z = w and not p ? a or b : not c
```

```
((((not x) = y) or (z = w)) and (not p)) ? (a or b) : (not c)
```




```
t1 := !c;
if a then t2 := a else t2 := b endif;
t3 := !x;
t4 := t3 == y;
if t4 then t5 := t4 else t5 := z == w endif;
if t5 then t6 := !p else t6 := t5 endif;
if t6 then return t2 else return t1 endif.
```

Esercizio 15

È data la seguente tabella di operatori relativi and un linguaggio **L**, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Associatività*	*Ordine valutazione*
`*`, **`and`**	sinistra	da sinistra a destra
`+`, **`or`**	destra	da destra a sinistra

e la seguente espressione:

```
x := a + b * c and d or f + g and h
```

Si assume che il linguaggio **L** manipoli unicamente tipi interi. In particolare, le operazioni booleane **and** ed **or** su tali interi sono definite dalle seguenti regole:

- Il risultato di un **and** è 1 se e solo se entrambi gli operandi sono diversi da 0, altrimenti il risultato è 0.
- Il risultato di un **or** è 0 se e solo se entrambi gli operandi sono uguali a 0, altrimenti il risultato è 1.

Si assume inoltre che gli operatori **and**, **or**, **+** e ***** siano valutati in corto circuito. In particolare, ecco le regole di corto circuito per i due operatori aritmetici:

- Se l'operando valutato per primo nella la somma (**+**) è 0, allora il risultato coincide con il secondo operando.
- Se l'operando valutato per primo nel prodotto (*****) è 1, allora il risultato coincide con il secondo operando.

Si chiede di:

a) **Rappresentare l'albero della espressione.**
b) **Specificare la semantica operazionale della espressione.**

NB: Il linguaggio di specifica operazionale è così caratterizzato:

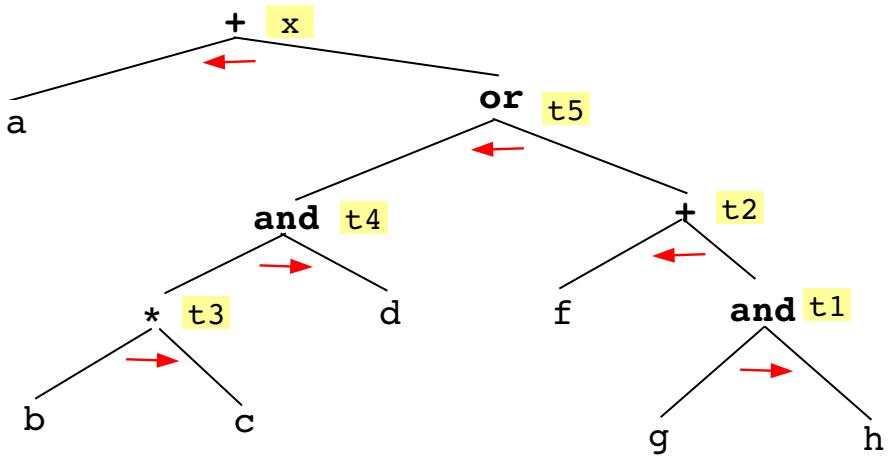
- Contiene gli operatori aritmetici (**+**, **-**, *****, */*), uguaglianza (**==**), disuguaglianza (**!=**) ed assegnamento (**=**);
- Non contiene gli operatori **and**, **or**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili o costanti;
- Contiene le istruzioni condizionali (*if-then-endif* ed *if-then-else-endif*) i cui predicati possono essere solo confronti tra una variabile ed una costante.

Esercizio 15

Operatore	*Associatività*	*Ordine valutazione*
*****, **and**	sinistra	da sinistra a destra
+, **or**	destra	da destra a sinistra

```
x := a + b * c and d or f + g and h
```

```
x := a + (((b * c) and d) or (f + (g and h)))
```




```
if g == 0 then t1 = 0 else t1 = h endif;
if t1 == 0 then t2 = f else t2 = t1 + f endif;
if t2 != 0 then
  t5 = 1
else
  if b == 1 then t3 = c else t3 = b * c endif;
  if t3 == 0 then t4 = 0 else t4 = d endif;
  t5 = t4;
endif;
if t5 == 0 then x = a else x = t5 + a endif.
```

Esercizio 16

È data la seguente tabella di operatori, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Tipo*	*Associatività*	*Ordine valutazione*	*Corto circuito*
`not`	unario	destra	-	-
`=`	binario	-	da sinistra a destra	no
`or`	binario	destra	da sinistra a destra	si
`and`	binario	sinistra	da destra a sinistra	no
`?:`	ternario	-	da sinistra a destra	si

e la seguente espressione: `a and b or not c = d ? x and y : x or y or z`

- Rappresentare l'albero della espressione.
- Specificare la semantica operazionale della espressione.

Il linguaggio di specifica operazionale è definito dalla seguente EBNF:

program → statements
statements → { stat **;** }$^{+}$
stat → assign-stat | if-stat | return-stat
assign-stat → **id :=** expr
expr → **id** | **!** **id** | **id == id** | **id && id**
if-stat → **if id then** statements { **elsif id then** statements } [**else** statements] **endif**
return-stat → **return id**

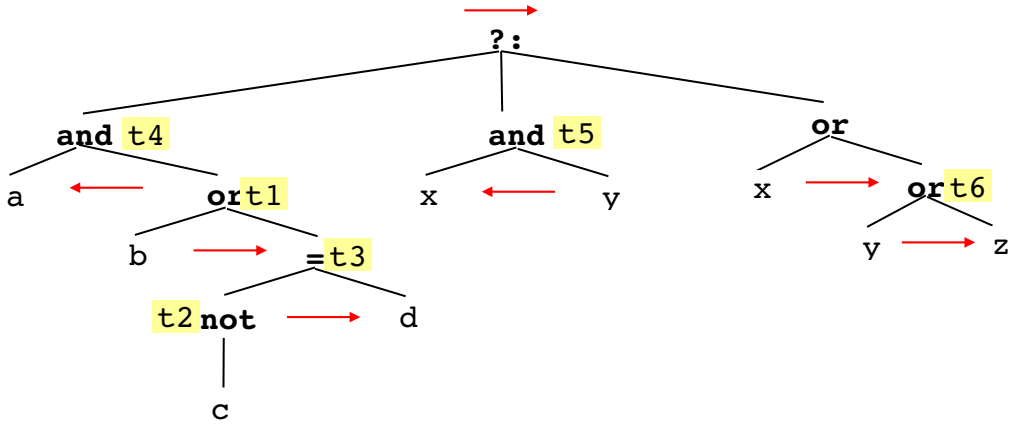
in cui **!** rappresenta la negazione logica, **&&** la congiunzione logica, **==** l'uguaglianza e **return** l'istruzione di terminazione del programma di specifica operazionale con restituzione del risultato.

Esercizio 16

Operatore	*Tipo*	*Assoc.*	*Ordine valutazione*	*Corto circuito*
not	unario	destra	-	-
=	binario	-	da sinistra a destra	no
or	binario	destra	da sinistra a destra	si
and	binario	sinistra	da destra a sinistra	no
?:	ternario	-	da sinistra a destra	si

```
a and b or not c = d ? x and y : x or y or z
```

```
(a and (b or ((not c) = d))) ? (x and y) : (x or (y or z))
```




```
if b then
  t1 := b;
else
  t2 := !c;
  t3 := t2 == d;
  t1 := t3;
endif;
t4 := t1 && a;
if t4 then
  t5 := y && x;
  return t5;
endif;
if x then
  return x;
endif;
if y then
  t6 := y;
else
  t6 := z;
endif;
return t6;
```

Esercizio 17

È data la seguente tabella degli operatori logici (per la quale si assume priorità decrescente dall'alto verso il basso), in cui tutti gli operatori binari sono valutati in corto circuito,

Operatore	*Descrizione*	*Tipo*	*Associatività*	*Ordine valutazione*
$\neg$	Negazione	unario	destra	-
$\vee$, $\wedge$	Disgiunzione, congiunzione	binario	sinistra	da destra a sinistra
$\Rightarrow$	Implicazione	binario	destra	da sinistra a destra

ed il seguente predicato: $a \wedge b \vee c \Rightarrow \neg x$

- Rappresentare l'albero del predicato.
- Specificare la semantica operazionale del predicato.

Il linguaggio di specifica operazionale è definito dalla seguente EBNF:

program → *stat-list*
stat-list → { *stat* **;** }$^+$
stat → *assign-stat* | *if-stat* | *return-stat*
assign-stat → **id :=** *expr*
expr → **true** | **false** | **id**
if-stat → **if id then** *stat-list* { **elsif id then** *stat-list* } [**else** *stat-list*] **endif**
return-stat → **return** *expr*

in cui **return** rappresenta l'istruzione di terminazione del programma di specifica operazionale con restituzione del risultato.

Esercizio 17

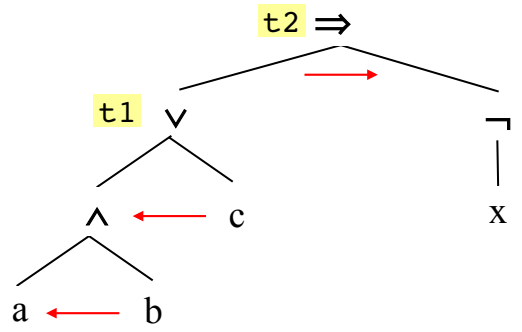
Operatore	*Descrizione*	*Tipo*	*Associatività*	*Ordine valutazione*
¬	Negazione	unario	destra	-
∨, ∧	Disgiunzione, congiunzione	binario	sinistra	da destra a sinistra
⇒	Implicazione	binario	destra	da sinistra a destra

$$a \wedge b \vee c \Rightarrow \neg x$$

$$((a \wedge b) \vee c) \Rightarrow (\neg x)$$

program → *stat-list*
stat-list → { *stat* **;** }$^{+}$
stat → *assign-stat* | *if-stat* | *return-stat*
assign-stat → **id :=** *expr*
expr → **true** | **false** | **id**
if-stat → **if id then** *stat-list* { **elsif id then** *stat-list* } [**else** *stat-list*] **endif**
return-stat → **return** *expr*

```
if c then
  t1 := true;
elsif b then
  t1 := a;
else
  t1 := false;
endif;
if t1 then
  if x then
    t2 := false;
  else
    t2 := true;
  endif;
else
  t2 := true;
endif;
return t2;
```



p	*q*	*p* ⇒ *q*
true	**true**	**true**
true	**false**	**false**
false	**true**	**true**
false	**false**	**true**

Corto circuito: $p \Rightarrow q \equiv (\, p \; ? \; q \; : \; \textbf{true} \,)$

Esercizio 18

È data la seguente tabella di operatori, per la quale si assume priorità decrescente dall'alto verso il basso,

Operatore	*Associatività*	*Ordine valutazione*
+,-,*, /	destra	da destra a sinistra
>, <, =	nonassoc	da sinistra a destra
and	sinistra	da destra a sinistra
or	destra	da destra a sinistra

e la seguente espressione:

```
a + b - c * d / f > (a - b) * z + w and x = y or x > y
```

Assumendo che l'operatore **and**, a differenza di **or**, sia valutato in corto circuito, si chiede di:

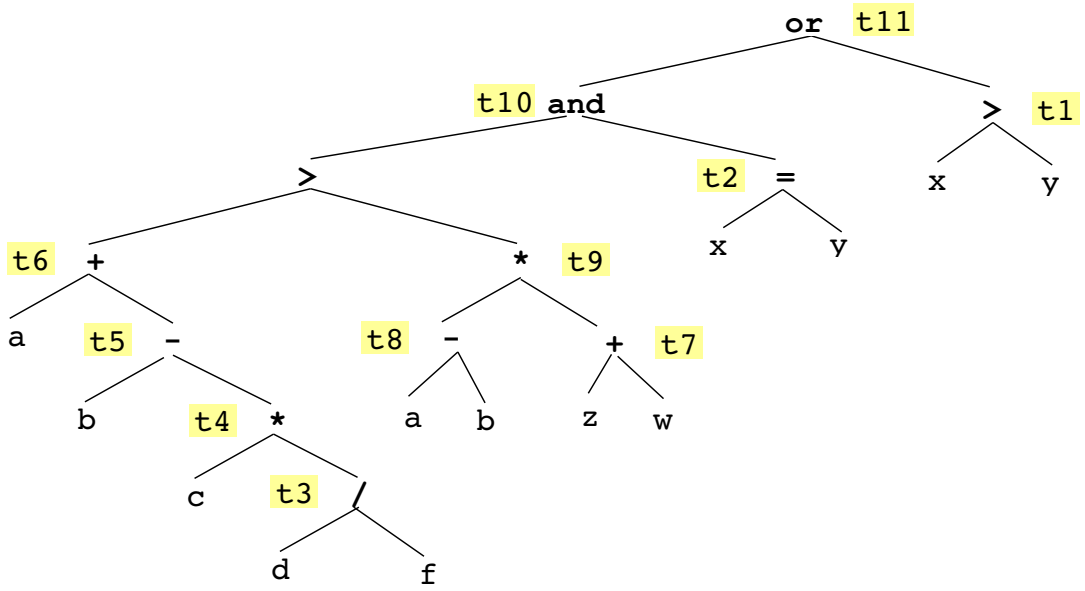
- Rappresentare l'albero della espressione;
- Specificare la semantica operazionale della espressione;
- Decorare l'albero della espressione con le variabili intermedie introdotte nella specifica operazionale.

Il linguaggio di specifica operazionale è così caratterizzato:

- Contiene l'operatore di assegnamento (**:**=), gli operatori aritmetici (+, **-**, *, /), gli operatori di confronto (>, <, =) e gli operatori logici (**&&**, **||**, **not**);
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili o costanti;
- Contiene le istruzioni condizionali (*if-then-endif* ed *if-then-else-endif*) e l'istruzione *return*, che restituisce il risultato e termina l'esecuzione.

Esercizio 18

```
a + b - c * d / f > (a - b) * z + w and x = y or x > y
```




```
t1 := x > y;
t2 := x = y;
if not t2 then
  t10 := false
else
  t3 := d / f;
  t4 := c * t3;
  t5 := b - t4;
  t6 := a + t5;
  t7 := z + w;
  t8 := a - b;
  t9 := t8 * t7;
  t10 := t6 > t9
endif;
t11 := t10 || t1;
return t11.
```