
Executar assignació, print, if i while

X29122_ca**INTRODUCCIÓ:**

L'entrada d'aquest exercici és una llista d'arbres binaris d'strings, on cadascun representa una instrucció d'un llenguatge de programació molt simple que descrivim a continuació.

Les variables del llenguatge poden guardar dígit, és a dir, valors entre 0 i 9. Les expressions del llenguatge tenen variables, dígit, i els operadors + i *, que treballen mòdul 10, de manera que el resultat d'avaluar un d'aquests operadors sobre dígit dona com a resultat un dígit. Per exemple, suposem que les variables *x* i *y* guarden valors 4 i 5, respectivament, i considerem l'expressió $3+x*2+y$. Llavors, l'avaluació d'aquesta expressió serà 6.

Nota: Quan una expressió conté una variable que no ha estat assignada abans, suposem que té valor per defecte 0.

Com a tipus d'instruccions del llenguatge, tenim assignació, *if*, *while*, amb el significat habitual, *print* per a escriure l'avaluació d'una expressió per la sortida seguida de salt de línia, i també subllistes d'instruccions. Per a les expressions de la condició de *if* i *while*, es considera que es compleixen si s'avaluen a diferent de 0, és a dir, a qualsevol dels valors 1,...,9. Aquest és un exemple de programa:

```
x = 1
while (x) {
    if (2*x) {
        print(x)
        print(3*x)
    }
    x = x + 1
}
print(x)
```

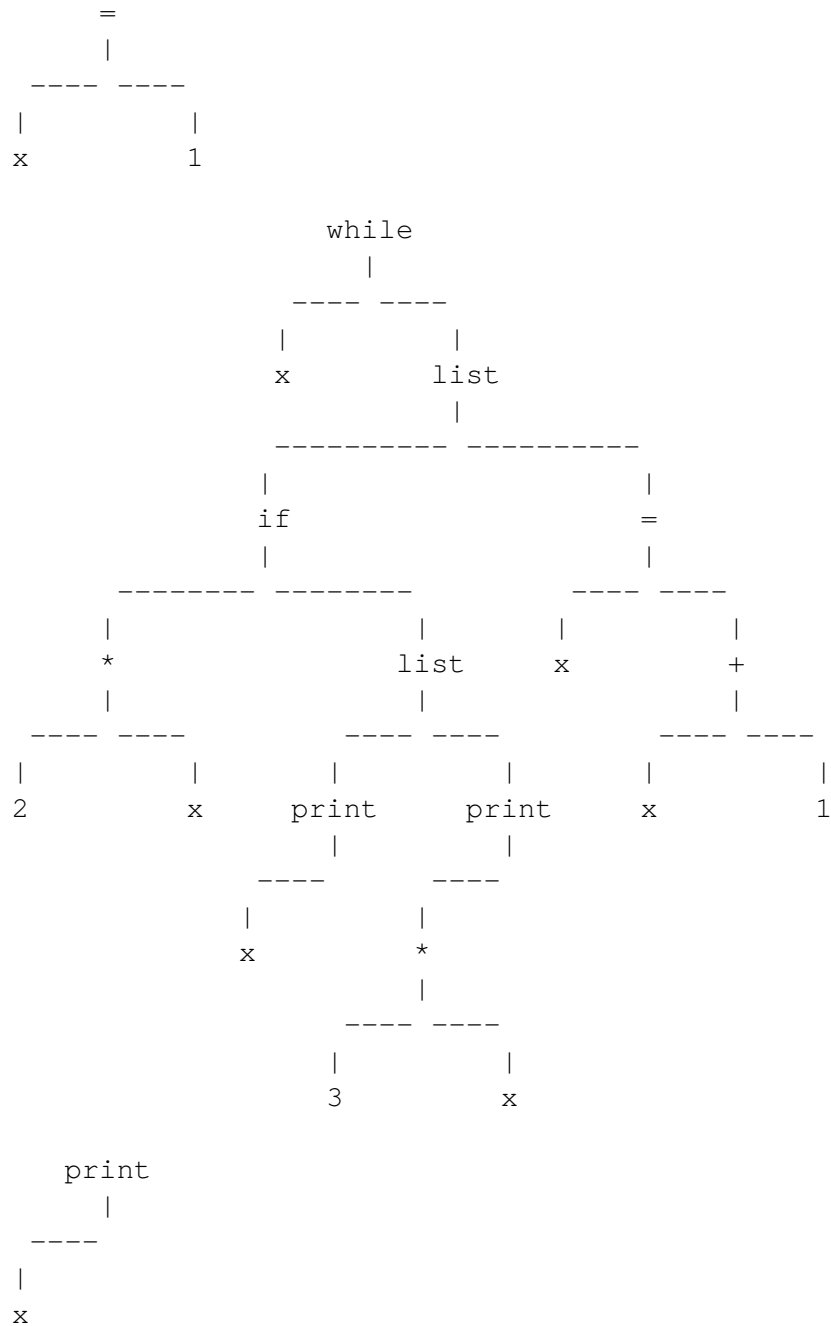
El programa anterior té com a sortida:

```
1
3
2
6
3
9
4
2
6
6
8
7
1
8
4
9
7
0
```

Com hem mencionat al principi, l'entrada d'aquest exercici és una llista d'arbres que representen instruccions d'aquest llenguatge. Per a l'exemple anterior de programa, aquesta seria l'entrada en INLINEFORMAT:

```
=(x, 1)
while(x, list(if(* (2, x), list(print(x), print(* (3, x))))), =(x, +(x, 1)))
print(x)
```

I aquesta seria la mateixa entrada en VISUALFORMAT:



L'objectiu d'aquest exercici és que implementeu dues funcions, una funció que avalua una expressió del llenguatge, i una funció que simula una instrucció del llenguatge. Convindrà que la segona funció cridi a la primera quan sigui convenient. Aquestes son les capceleres:

```
// Pre: t és un arbre no buit d'strings que representa una expressió correcta
//       sobre díigits i variables que guarden díigits, i els operadors +,* mòdul
//       En particular, l'arrel de t és o bé +, o bé *, o bé un dígit, o bé una
//       var2val és un mapeig de variables a díigits.
// Post: Retorna l'avaluació de l'expressió representada per t reemplaçant les
//       pels seus corresponents valors definits a var2val, o per 0 si no estan
int evaluate(map<string,int> &var2val, BinTree<string> t);

// Pre: t és un arbre no buit d'strings que representa una instrucció correcta
//       del llenguatge de programació descrit a l'enunciat.
//       En particular, o bé és l'arbre buit,
//       o bé la seva arrel és, o bé =, o bé print, o bé if, o bé while,
//       o bé list, cas en el qual, representa una subllista d'instruccions.
// Post: S'ha simulat l'execució d'aquesta instrucció, modificant var2val
//       i escrivint el que calgui per la sortida estandar,
//       d'acord a aquesta simulació.
void execute(map<string,int> &var2val, BinTree<string> t);
```

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix uns fitxers que haureu d'utilitzar: `main.cc`, `BinTree.hh`, `execute.hh`. Us falta crear el fitxer `execute.cc` amb els corresponents `includes` i implementar-hi les dues funcions anteriors. Només cal que pugueu `execute.cc` al jutge.

Entrada

La primera línia de l'entrada descriu el format en el que es descriuen els arbres, o bé `IN-LINEFORMAT` o bé `VISUALFORMAT`. Després venen un nombre arbitrari d'instruccions representades per arbres. Fixeu-vos en que el programa que us oferim ja s'encarrega de llegir aquestes instruccions i mantenir un mapeig de variables a díigits amb el valor actual de les variables. Només cal que implementeu les dues funcions abans esmentades.

Nota: Els casos d'entrada s'han creat mirant de garantir que l'execució de les instruccions d'entrada acaben.

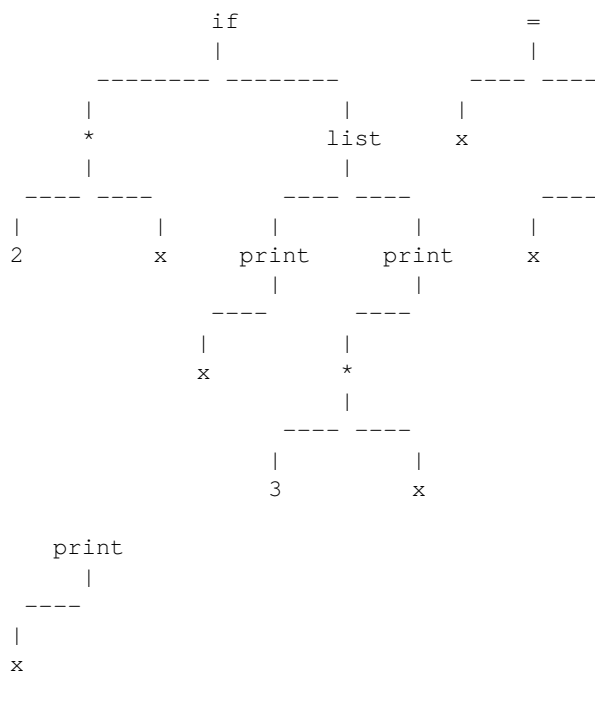
Sortida

La sortida conté el que seria la sortida esperada com a resultat d'executar les instruccions que venen descrites a l'entrada. Fixeu-vos en que el programa que us oferim ja s'encarrega de cridar a la vostra funció `execute` que alhora hauria de cridar a la funció `evaluate` quan calgui. Només cal que implementeu aquestes funcions de manera que simulin correctament el comportament de les instruccions, actualitzant el mapeig de variables a díigits correctament, i escrivint díigits per la sortida amb la instrucció `print`. Cadascun d'aquests díigits escrits ha de venir seguit d'un salt de línia.

Exemple d'entrada 1

```
VISUALFORMAT
  =
  |
  -----
 |           |
x             1
```

```
while
  |
  -----
 |           |
x             list
              |
  -----
 |           |
```



Exemple de sortida 1

```

1
3
2
+
6
3
9
4
2
6
8
7
1
8
4
9
7
0

```

Exemple d'entrada 2

```

INLINEFORMAT
=(x,1)
while(x,list(if(* (2,x),list(print(x),print(* (3,x),)),=(x,+(x,1))))
print(x,)
```

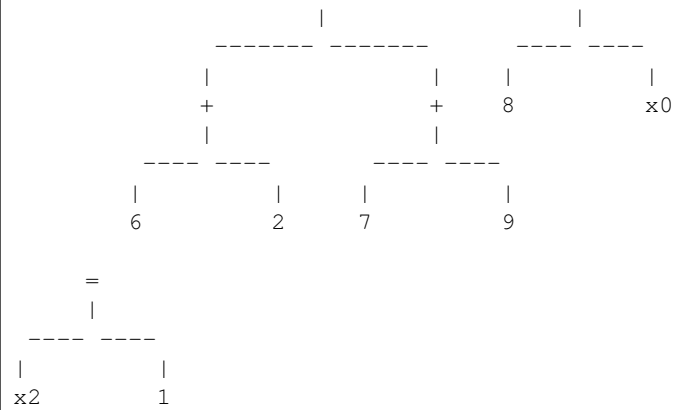
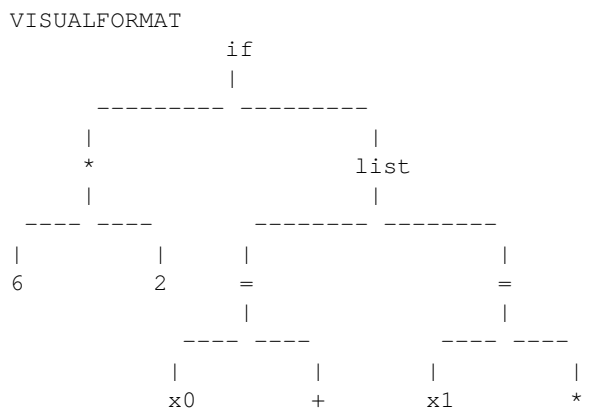
Exemple de sortida 2

```

1
3
6
3
9
4
2
6
8
7
1
8
4
9
7
0

```

Exemple d'entrada 3





Exemple de sortida 4

```

8
1 * (8, x0)))
5
2), ), list (print (x2, ), list (print (5, ), =(x2, +(x2, 1))))))
2 +(4, x3), 8), x3), list (print (2, ), =(x3, +(x3, 1))))), =(x3, *
5 print (+(+(x3, x0), 1), ), list (=(x3, *(4, x0)), =(x2, +(x0, x3))
8(x3, )))
3
5 =(x5, +(x5, 1)))
3
4
5
9
7
2
4
6

```

Entenem com a solució ràpida una que és correcta, de cost lineal i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autoria: PRO2

Generació: 2026-01-25T14:56:28.562Z

© *Jutge.org*, 2006–2026.

<https://jutge.org>