Control LP (Compiladors): Muntanyes!

Entregueu un únic arxiu amb el nom mountains.g

Cal fer un compilador per interpretar un llenguatge simple per definir muntanyes. Es poden declarar muntanyes de forma literal, utilitzant símbols per definir pujada (/), cim (-) i baixada (\), combinats amb l'operador de concatenació (;). També es poden definir muntanyes a partir d'altres muntanyes. Les definicions de les muntanyes hauran de tenir com a mínim un símbol de cada tipus (pujada, cim i baixada), i per a ser considerades correctes, definicions de muntanyes en seqüència (i.e., de serralades) hauran de consistir d'aquests tres símbols repetits tantes vegades com es vulgui. Existeixen instruccions per completar (quan sigui possible) una definició de muntanya que li falta algun dels símbols a la seva definició, així com les instruccions condicionals i de bucle. També existeixen funcions definides per definir pics i valls, així com per determinar si el punt més alt de dues muntanyes està a la mateixa alçada. Finalment, es poden també definir variables numériques per a utilitzar en algunes de les funcions comentades anteriorment. A continuació podeu veure un exemple complet d'aquest llenguatge.

```
M1 is 1*/;2*-;3*\                       // Declarem una muntanya amb 1 unitat de pujada, 2 de cim i 3 de baixada
M2 is 3*/;4*-;3*\
M12 is #M1;#M2                          // Declarem una muntanya a partir d'altres dues
M3 is #M2;1*/;2*-;3*\;1*/;1*-;1*\

M4 is Peak(2,4,2)                       // Declarem un pic amb 2 unitats de pujada, 4 de cim i 2 de baixada
M5 is Valley(3,4,3)                     // Declarem una vall amb 3 unitats de baixada, 4 de cim i 3 de pujada

if (Match(#M4,#M5) OR Height(#M12) == 6)  // Condicional utilitzant funcions de igualtat d'alçada i
  M6 is #M4;#M5                         // verificació que l'alçada d'una de les muntanyes es 6
  Draw(#M6)                             // Dibuixem la muntanya
endif

Draw(1*/;2*-;3*\)

M is 1*/                                // Definició de muntanya incompleta: falten dos simbols.

if (NOT Wellformed(M))                  // Funció que detecta si la muntanya esta incompleta
  Complete(M)                           // Instruccio que completa (afegeix la part que falta)
endif

k is 1                                  // Variable numerica

while (Height(#M2) < 10 AND Match(#M4,#M5)) // Bucle
 M7 is #M7;Peak(k,k+1,k)
 Draw(#M7)
endwhile
```

Figura 1: Llenguatge per definir muntanyes.

Assumiu que com a condicions dels condicionals i bucles només poden aparèixer els operadors relacionals (<, >) i el de igualtat, que poden ser combinats amb els operadors `AND`, `OR` i `NOT` com es mostra a l'exemple, amb la prioritat típica (operadors relacionals més prioritaris que `AND`, que és més prioritari que `OR`, que és més prioritari que `NOT`).

[Part 1.] Defineix la part lèxica i sintàctica. Fès la gramàtica per a que PCCTS pugui reconèixer-la i decorar-la per generar l'AST mostrat a continuació.

```
list
 \__is
 |    \__id(M1)
 |    \__;
 |        \__;
 |        |   \__*
 |        |   |   \__intconst(1)
 |        |   |   \__/
 |        |   \__*
 |        |       \__intconst(2)
 |        |       \__-
 |        \__*
 |            \__intconst(3)
 |            \__\
 \__is
 |    \__id(M2)
 |    \__;
 |        \__;
 |        |   \__*
 |        |   |   \__intconst(3)
 |        |   |   \__/
 |        |   \__*
 |        |       \__intconst(4)
 |        |       \__-
 |        \__*
 |            \__intconst(3)
 |            \__\
 \__is
 |    \__id(M12)
 |    \__;
 |        \__id(M1)
 |        \__id(M2)
 \__is
 |    \__id(M3)
 |    \__;
 |        \__;
 |        |   \__;
 |        |   |   \__;
 |        |   |   |   \__;
 |        |   |   |   |   \__;
 |        |   |   |   |   |   \__id(M2)
 |        |   |   |   |   |   \__*
 |        |   |   |   |   |       \__intconst(1)
 |        |   |   |   |   |       \__/
 |        |   |   |   |   \__*
 |        |   |   |   |       \__intconst(2)
 |        |   |   |   |       \__-
 |        |   |   |   \__*
 |        |   |   |       \__intconst(3)
 |        |   |   |       \__\
 |        |   |   \__*
 |        |   |       \__intconst(1)
 |        |   |       \__/
 |        |   \__*
 |        |       \__intconst(1)
 |        |       \__-
 |        \__*
 |            \__intconst(1)
 |            \__\
 \__is
```

```
|    \__id(M4)
|    \__Peak
|           \__intconst(2)
|           \__intconst(4)
|           \__intconst(2)
\__is
|    \__id(M5)
|    \__Valley
|             \__intconst(3)
|             \__intconst(4)
|             \__intconst(3)
\__if
|    \__OR
|    |    \__Match
|    |    |       \__id(M4)
|    |    |       \__id(M5)
|    |    \__==
|    |         \__Height
|    |         |       \__id(M12)
|    |         \__intconst(6)
|    \__list
|           \__is
|           |    \__id(M6)
|           |    \__;
|           |        \__id(M4)
|           |        \__id(M5)
|           \__Draw
|                  \__id(M6)
\__Draw
|      \__;
|          \__;
|          |   \__*
|          |   |   \__intconst(1)
|          |   |   \__/
|          |   \__*
|          |       \__intconst(2)
|          |       \__-
|          \__*
|              \__intconst(3)
|              \__\
\__is
|    \__id(M)
|    \__*
|        \__intconst(1)
|        \__/
\__if
|    \__NOT
|    |     \__Wellformed
|    |                 \__id(M)
|    \__list
|           \__Complete
|                     \__id(M)
\__is
|    \__id(k)
|    \__intconst(1)
\__while
        \__AND
        |     \__<
        |     |   \__Height
        |     |   |       \__id(M2)
        |     |   \__intconst(10)
        |     \__Match
        |             \__id(M4)
        |             \__id(M5)
        \__list
```

```
\__is
|    \__id(M7)
|    \__;
|        \__id(M7)
|        \__Peak
|               \__id(k)
|               \__+
|               |   \__id(k)
|               |   \__intconst(1)
|               \__id(k)
\__Draw
       \__id(M7)
```

[Part 2.] Interpretació.