Pràtica LP (Compiladors): anem a fer compres!

Cal fer un compilador per interpretar una comanda de compres. Es podran definir compres de forma literal, definits amb una llista de parells (quantitat, producte), així com combinar compres definides prèviament per definir una de nova, amb els operadors `AND`, `MINUS` i `*`. El següent exemple mostra el llenguatge:

```
Compres1 = [(1,mantega),(2,tomaquets),(20,aigua)]   // Compra de 3 productes
Compres2 = [(2,pomes)]                              // Compra de 2 pomes
Compres3 = Compres1 AND Compres2                    // Compres3 conte la unio
Compres4 = [(1,tomaquets)]
Compres5 = Compres3 MINUS Compres4                  // Compres5 conte la resta
Compres6 = [(1,dentrific)]
Compres7 = 3*Compres5 AND Compres6                  // Us multiplicacio compres
UNITATS Compres7                                    // Retorna 73
Compres8 = 3*(Compres3 MINUS Compres2)
UNITATS Compres8                                    // Retorna 69
DESVIACIO Compres7                                  // Desviacio standard Compres7
PRODUCTES Compres7                                  // Llista productes Compres7
```

S'assumeix que les llistes estan semànticament ben formades en quant a l'ús dels operadors, és a dir:

- Les llistes usades a la dreta d'una assignació estan definides en algun moment.
- L'operador `MINUS` és binari, i s'assumeix que es restaran productes del primer segon operand al primer operand. Per cada producte, la quantitat del segon operand ha de ser menor o igual a les del mateix producte del primer operand (no cal que ho comproveu!). Per exemple, la definició de `Compres5` és correcta ja que es possible treure un tomaquet de `Compres3` (indicat a `Compres4`).
- L'operador `*` és binari, i l'operand de l'esquerre és sempre un natural.

Defineix la part lèxica (tokens) i sintàctica (gramàtica). En quant als operadors, la precedència és: `()` > `*` > (`MINUS`,`AND`). Fès la gramàtica per a que PCCTS pugui reconèixer-la i decorar-la per generar l'AST mostrat a l'anvers de la pàgina .
La regla inicial de la gramàtica és:

```
compres: (list_comps)* <<createASTlist();>> ;
```

Fés els mètodes

```
int unitats(AST *a)
double stdev(AST *a)
int productes(AST *a)
```

on `a` apunta a un node etiquetat `UNITATS`, `DESVIACIO` i `PRODUCTES`, respectivament. En el codi que se't facilita ja hi ha definida la funció:

`AST *findASTCompraDef(string id)`

que donat un nom d'una compra, retorna el node de l'AST on està definida. Al programa principal, has de recòrrer l'AST i cada cop que trobis una de les funcions anteriors has de cridar a la funció corresponent per a que l'avalui.

```
list
 \__=
 |   \__id(Compres1)
 |   \__[
 |       \__1
 |       |   \__id(mantega)
 |       \__2
 |       |   \__id(tomaquets)
 |       \__20
 |            \__id(aigua)
 \__=
 |   \__id(Compres2)
 |   \__[
 |       \__2
 |           \__id(pomes)
 \__=
 |   \__id(Compres3)
 |   \__AND
 |         \__id(Compres1)
 |         \__id(Compres2)
 \__=
 |   \__id(Compres4)
 |   \__[
 |       \__1
 |           \__id(tomaquets)
 \__=
 |   \__id(Compres5)
 |   \__MINUS
 |           \__id(Compres3)

 |           \__id(Compres4)
 \__=
 |   \__id(Compres6)
 |   \__[
 |       \__1
 |           \__id(dentrific)
 \__=
 |   \__id(Compres7)
 |   \__AND
 |         \__*
 |         |   \__3
 |         |   \__id(Compres5)
 |         \__id(Compres6)
 \__UNITATS
 |         \__id(Compres7)
 \__=
 |   \__id(Compres8)
 |   \__*
 |       \__3
 |       \__MINUS
 |               \__id(Compres3)
 |               \__id(Compres2)
 \__UNITATS
 |         \__id(Compres8)
 \__DESVIACIO
 |           \__id(Compres7)
 \__PRODUCTES
             \__id(Compres7)
```