Control LP (Compiladors): Karel

Cal fer un compilador per interpretar una variant del llenguatge per controlar el robot Karel. El llenguatge té instruccions per definir l'entorn on es mourà el robot (dimensions, on hi ha parets i sensors, la posició inicial), moure's en el sentit que està orientat, girar a l'esquerra, preguntar si hi ha parets en la direcció de moviment, si hi ha sensors per recollir, executar condicionalment instruccions, repetir un cert nombre de vegades instruccions, executar subfuncions definides i apagar-se. Per indicar l'orientació i els costats de cada cel·la de l'entorn usarem left, right, up i down. El següent exemple mostra el llenguatge:

```
world 5 5                          //defineix les dimensions del mon
robot 2 3 5 right                  //posició, nombre de sensors i orientació inicial del robot
walls [2 2 left, 1 2 up]           //indica una llista de paret: cel.la i orientació
beepers 1 3 2                      //indica que hi ha 2 sensor a la cel.la (1,3)
walls [1 2 down]
beepers 4 4 1

define turnright {                 //defineix la subfunció turnright
 turnleft;                         //canvia l'orientació
 turnleft;
 turnleft;
}

define t1 {
 turnright;
 if isClear and anyBeepersInBag {  //condició: no te paret en la seva orientació i li queden sensors
  move;                            //es mou una cel.la en el sentit que està orientat
  putbeeper;                       //afegeix un sensor a la cel.la on es troba
 }
}

begin
iterate 3 {                        //repeteix 3 cops les instruccions
 if foundBeeper { pickbeeper; }    //condició: hi ha algun sensor a la cel.la; acció: l'agafa
 if not foundBeeper {
  move; turnright;                 //es mou i executa la subfunció turnright
 }
}
t1;                                //executa la subfunció turnright
turnoff;                           //apaga el robot
end
```

S'assumeix que inicialment esta la definició del mon i la posicio inicial del robot seguit de les definicions de parets, sensors i definicions de subfuncions i finalment la llista d'instruccions entre el begin i l'end. Noteu que el `move` només canvia la posició si es pot realitzar (és a dir, no hi ha paret i no ens sortim del mon). La resta d'operacions bàsiques no canvien la posició. El `turnleft` canvia l'orientació a l'esquerra, és a dir, de `down` a `right`, de `right` a `up`,...

Heu de considerar que l'and i l'or associen a l'esquerra i tenen la mateixa prioritat.

[Part 1: 50% nota] 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 l'anvers de la pàgina.
[Part 2: 50% nota] Interpretació: fes el mètode

```
void novaPosicio(AST *a)
```

on `a` apunta al node arrel de l'arbre. La funció interpretarà el programa en temps real per determinar la nova posició del robot, és a dir, s'imprimirà per la sortida estàndard la posició del robot després d'executar el programa partint de la posició inicial definida al començament. En el codi que has de fer assumeix que ja hi ha definida la funció:

```
AST *findDefinition(string id)
```

que donat el nom d'una subfunció, retorna el node de l'AST on està definida. També teniu les funcions

```
bool evaluateCondition(AST *a)
```

on `a` apunta a la condició d'un `if` i ens retorna el resultat d'avaluar-la; i

```
bool dinsDominis (int x, int y)
bool isClear (int x, int y, int orient)
```

que ens indiquen respectivament si la posició `(x,y)` és dins dels dominis del mon i si en la posició inicada i amb l'orientació donada no hi ha cap paret (podeu assumir que la funció va bé amb la codificació de l'orientació que hagiu triat).

```
list
 \__world
 |       \__5
 |       \__5
 \__robot
 |       \__2
 |       \__3
 |       \__5
 |       \__right
 \__list
 |      \__walls
 |      |       \__2
 |      |       \__2
 |      |       \__left
 |      |       \__1
 |      |       \__2
 |      |       \__up
 |      \__beepers
 |      |         \__1
 |      |         \__3
 |      |         \__2
 |      \__walls
 |      |       \__1
 |      |       \__2
 |      |       \__down
 |      \__beepers
 |      |         \__4
 |      |         \__4
 |      |         \__1
 |      \__define
 |      |        \__id(turnright)
 |      |        \__list
 |      |               \__turnleft
 |      |               \__turnleft
 |      |               \__turnleft
 |      \__define
 |               \__id(t1)
 |               \__list
 |                      \__id(turnright)
 |                      \__if
 |                           \__and
 |                           |     \__isClear
 |                           |     \__anyBeepersInBag
 |                           \__list
 |                                  \__move
 |                                  \__putbeeper
```

```
\__list
       \__iterate
       |         \__3
       |         \__list
       |                \__if
       |                |    \__foundBeeper
       |                |    \__list
       |                |           \__pickbeeper
       |                \__if
       |                     \__not
       |                     \__foundBeeper
       |                     \__list
       |                            \__move
       |                            \__id(turnright)
       \__id(t1)
       \__turnoff
```