Linguaggi e Traduttori
a.a. 2005/2006
Tema d'esame del 7 Novembre 2006

Si costruisca, utilizzando la coppia di programmi JFLEX e CUP, un traduttore guidato dalla sintassi in grado di riconoscere un linguaggio che permette di descrivere il percorso di un veicolo all'interno di uno spazio bidimensionale.

Linguaggio di ingresso

Il file che descrive il percorso inizia con una sezione contenente i seguenti 3 campi:

1. Un campo alfanumerico di tipo stringa di 8 caratteri, iniziante con un carattere maiuscolo e contente un codice identificativo del veicolo.
2. La data corrente nel formato ***gg/mm/aaaa***
3. Una descrizione testuale delimitata da **<*!*** ….. ***!*>**

Il codice del veicolo e la data sono sempre presenti, mentre la descrizione può essere opzionale. L'ordine in cui devono apparire nel file di descrizione è quello dell'elenco numerato.

La seconda sezione è separate dalla prima dai caratteri ***"%%"*** e contiene le specifiche sul percorso. Il linguaggio che descrive l'itinerario contiene un set di primitive che permettono di definire come il veicolo deve muoversi:

- **Angle(α):** ruota il veicolo finché è orientato nella direzione che forma un angolo α rispetto all'asse delle *x* (analizzato in senso orario). α può essere specificato sia in radianti che in gradi ed è un numero con segno; in radianti l'angolo è rappresentato da una frazione o da un numero reale seguito dal simbolo '***PI***' (es. ***3/4PI*** o ***-0.5PI)*** mentre in radianti è semplicemente rappresentato con un numero reale.
- **Move(k T):** muove il veicolo lungo la direzione corrente. **T** può rappresentare sia un distanza di **k** metri (**T** è *'M'* o *'m'*) o un tempo di **k** secondi (**T** è *'S'* o *'s'*). **k** è un numero reale positivo.
- **Speed (k):** setta la velocità del veicolo a **k** metri al secondo, **k** è un numero reale positivo.
- **AcquireData (k):** ferma il veicolo per **k** secondi per acquisire dei dati.

Le primitive sono separate dal simbolo ***';'*** e formano una sequenza di comandi specificabili in un qualsiasi ordine ad eccezione del commando **Speed(k)** che deve essere specificato come primo comando della sequenza e dopo ogni comando **AcquireData**.

Il linguaggio permette la definizione di cicli **for** al fine di ripetere un certo numero di volte un set di comandi racchiusi nel blocco delimitato dai simboli ***'['*** e ***']'*** . Il ciclo for è espresso come:

- **for (u)[**

  ……..

  ]

dove **u** è un numero intero positivo diverso da 0 che specifica il numero di volte che devono essere ripetuti i comandi racchiusi nel blocco del ciclo. All'interno di un blocco for, il primo comando deve essere Speed.

Il linguaggio prevede anche un costrutto di condizione che esegue il codice all'interno del blocco delimitato dai simboli ***'['*** e ***']'*** se una determinata condizione è vera. Nel linguaggio sono previste condizioni sulla distanza del veicolo dall'origine e sul tempo totale trascorso dalla sua partenza. Sono supportati gli operatori relazionali **'>'** e **'<'**. Il costrutto è espresso come:

- **if (T relOp k) [**

  ……

  ]

dove **k** è un numero reale e **T** può essere *'S'* or*'s'* (per rappresentare il tempo trascorso) o *'M'* or *'m'* (per rappresentare la distanza dall'origine).

Il linguaggio può perciò contenere 3 tipi di blocchi (in ogni ordine e in qualsiasi numero): lista di comandi semplici, costrutti for e costrutti if. I costrutti **for** e **if** non possono essere annidati (*non possono esistere costrutti **for** o **if** all'interno di costrutti **for** o **if***).

Scopo del programma

Il programma deve verificare la correttezza sia sintattica che grammaticale del file di ingresso e, in caso positivo, calcolare la posizione finale del veicolo rispetto all'origine (x=0, y=0) e il tempo totale trascorso dalla partenza. E' consentito l'utilizzo di variabili globali solo per memorizzare la velocità e l'angolo corrente (Si usino gli attributi sintetizzati e l'oggetto predefinito RESULT per propagare la posizione corrente e il tempo trascorso nell'albero di derivazione).

NB: Nota che gli spostamenti DeltaX e DeltaY relativi a un segmento di lunghezza L formante un angolo Alpha rispetto all'asse delle *X* possono essere ottenuti dalla formula:

deltaX = k * cos(alpha) deltaY = k * sen(alpha)

A questo proposito java fornisce dei metodi statici della classe Math: java.Math.sin e java.Math.cos (*e la costante java.Math.PI)* che, dato un double in ingresso, ritornano un risultato di tipo double.

Example

```
AX23FF34
02/11/2006
<! Route B !>
%%
Speed( 3.5 );
Move( 10 M );
for( 4 )[
     Speed( 3);
     Angle(-2/3PI);
     Move(12.5 S);
]
if( s > 3.5)[
     AcquireData( 34 );
     Speed(1);
]
Angle(0);
Move( 3 S);
```

INPUT

OUTPUT

```
Syntax correct
The final position is X = 34.5,
Y = -12.56. The total time
elapsed is 134.5 sec
```