Linguaggi Formali e Compilatori
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
20/06/2011

COGNOME: .....
NOME: ..... Matricola: .....
Iscritto a: ○ Laurea Specialistica ○ V. O. ○ Laurea Triennale ○ Altro: ...
Sezione: ○ Prof. Breveglieri ○ Prof. Crespi ○ Prof.Morzenti

Per la risoluzione della domanda relativa alle esercitazioni si deve utilizzare l'implementazione del compilatore `Acse` che viene fornita insieme al compito.

Si richiede di modificare la specifica dell'analizzatore lessicale da fornire a `flex`, quella dell'analizzatore sintattico da fornire a `bison` ed i file sorgenti per cui si ritengono necessarie delle modifiche in modo da estendere il compilatore `Acse` con la possibilità di gestire gli operatori *ship* e *in*.

```
1 a = b <=> c;
```
(a) Operatore *ship*

```
1 a = b in c:d;
```
(b) Operatore *in*

Figura 1: Esempi di operatori *ship* ed *in*

L'operatore *ship* permette di confrontare due espressioni. La semantica dell'espressione `a <=> b` è definita dalla funzione $ship(a, b)$:

$$ship(a,b) = \begin{cases} -1 & se \quad a < b \\ 0 & se \quad a = b \\ 1 & se \quad a > b \end{cases}$$

L'operatore *in* è anch'esso un operatore di confronto e permette di verificare se il valore di una espressione è contenuto all'interno di un intervallo. La semantica associata all'espressione `a in b:c` è definita dalla funzione $in(a, b, c)$:

$$in(a,b,c) = \begin{cases} 0 & se \quad a < b \vee a > c \\ 1 & se \quad b \leq a \leq c \end{cases}$$

Per semplicità si assumano che i bound b, c siano ordinati, cioè sia sempre vero che $b \leq c$.

Si espliciti ogni eventuale ulteriore assunzione che sia ritenuta necessaria a completare la specifica data.

[1] Tempo 45'. Libri e appunti personali possono essere consultati.
È consentito scrivere a matita. Scrivere il proprio nome sugli eventuali fogli aggiuntivi.

1. Definire i token (e le relative dichiarazioni in `Acse.lex` e `Acse.y`) necessari per ottenere la funzionalità richiesta. (3 punti)

   La soluzione è riportata nella patch allegata.

2. Definire le regole sintattiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (4 punti)

   La soluzione è riportata nella patch allegata.

3. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (18 punti)

   La soluzione è riportata nella patch allegata.

4. Dato il codice di Figura 2:

```
1  a = a + b << c + d;
```

Figura 2: Addizioni e scorrimenti

Scrivere l'albero sintattico relativo partendo dalla grammatica Bison definita in `Acse.y` *iniziando dal non-terminale* `exp` a precedenza più bassa. (5 punti)

La soluzione è riportata in Figura 3.

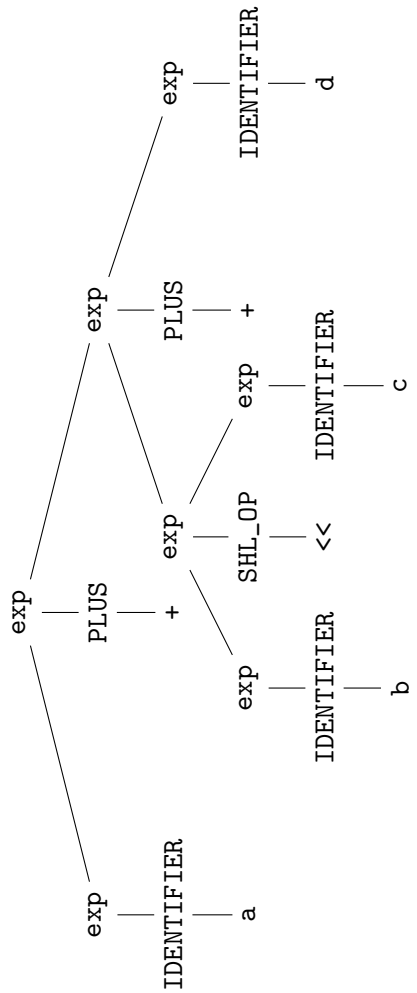

Figura 3: Albero sintattico del codice riportato in Figura 2

5. (Bonus) Si supponga di voler implementare una versione dell'operatore *ship* di modo che supporti il confronto tra array.

   - Che semantica avrebbe tale operatore?
   - Come dovrebbe essere implementato?
   - Esiste un costrutto simile a voi noto?

   L'operatore *ship* esteso agli array non è altro che la funzione `strcmp` della libreria C delle stringhe. Essa confronta due stringhe (a.k.a array di caratteri), mentre l'operatore *ship* esteso supporta array di interi:

   $$ship(A,B) = \begin{cases} -1 & se \quad \exists i \mid pref(A,i) = pref(B,i) \wedge A_i < B_i \\ 0 & se \quad A = B \\ 1 & altrimenti \end{cases}$$

   Dove la funzione $pref(A,i)$ restituisce il prefisso di lunghezza i del vettore A.

   L'implementazione è banale. Basta generare del codice che iteri sull'array A fino a quando l'elemento A_i è uguale all'elemento B_i. Se per un j si trovano due elementi A_j, B_j tali che $A_j < B_j$ allora l'array A è "più piccolo" dell'array B. Se tale elemento non viene trovato, oppure si raggiunge la fine dell'array B allora l'array A è "più grande" dell'array B. Ovviamente, se i due array sono identici, l'operatore *ship* ritorna 0.

Applicare una patch

Sul sito del corso è disponibile una patch contenente la soluzione del tema d'esame per quanto riguarda la modifica della macchina `Acse`.

Per applicare la patch:

1. scaricare la macchina `Acse` versione 1.1.0
2. scaricare la patch `soluzione-20-06-11.diff`
3. scompattare l'archivio contenente la macchina `Acse`
4. usando il terminale, portarsi nella directory in cui è stata estratta la macchina `Acse`
5. copiare in tale cartella la patch
6. applicare la patch tramite il comando

```
patch -p1 < soluzione-20-06-11.diff
```

La patch è un normalissimo file di testo, contenente le differenze tra la versione di `Acse` con implementata la soluzione dell'esame e la versione 1.1.0.

Le righe che iniziano con il carattere `+` sono state aggiunte, mentre quelle con il carattere `-` sono state rimosse.