Linguaggi Formali e Compilatori
Proff. Breveglieri, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
11/02/2014

COGNOME: .....
NOME: ..... Matricola: .....
Corso: ∘ Laurea Specialistica ∘ V. O. ∘ Laurea Triennale ∘ Altro: ...
Sezione: ∘ Prof. Breveglieri ∘ 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 espressioni condizionali Python-like. Si consideri il seguente esempio:

```
int a;
int b;
read(a);
read(b);
b = a * 2 - b if a > 2 * b else 0;
write(b);
```

L'espressione condizionale qui presentata è sempre definita per ogni valore della condizione, indicata tra le keyword `if` ed `else`, e vale l'espressione che precede la keyword `if` nel caso la condizione sia vera, altrimenti l'espressione che segue la keyword `else`. Si specifichi inoltre la precedenza e associatività degli operatori in modo opportuno al fine di garantire la composizione di espressioni condizionali.

Sono *preferibili* implementazioni del costrutto che non mantengano il flusso di esecuzione lineare.

[1] Tempo 60'. 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. (2 punti)

2. Definire le regole sintattiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (3 punti)

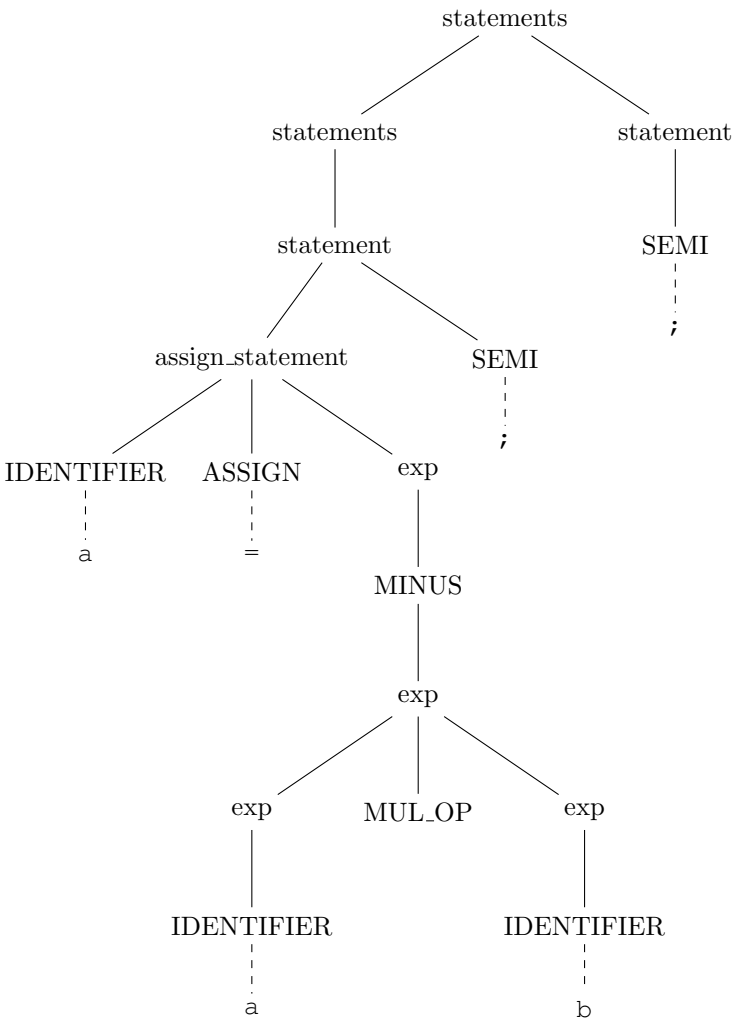
3. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (20 punti)

   La soluzione è riportata nella patch allegata.

4. Data il seguente snippet di codice `Lance`:

```
a = -a * b;;
```

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



5. (**Bonus**) Si descriva come estendere il costrutto dato in modo da poter specificare un valore alternativo per l'espressione qualora questa abbia valore zero.

```
x = a if c > 2 else b onzero 42;
```