Linguaggi Formali e Compilatori
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
25/09/2013

COGNOME: .....
NOME: ..... Matricola: .....
Corso: ○ 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 il costrutto *loop-decreasing*.

```
int x[100], input, c = 0, s = 100;

loop_decreasing s by c  {
  read(input);
  c = c + 5;
  write(c);
} while (input > 32 && c < 50);

write(input);
write(s);
```

Figura 1: Esempio

Questo tipo di **ciclo** ha come parametri un *variabile contatore*, un *espressione di decremento* e una *condizione di esecuzione*. Il costrutto ha i seguenti vincoli:

- condizione necessaria per l'esecuzione del ciclo è che la variabile di conteggio sia **positiva**.
- l'esecuzione di tutte le iterazioni **ad eccezione della prima** è controllata anche dalla *condizione di esecuzione*: se la condizione è falsa il flusso di controllo esce dal ciclo. In altre parole, la condizione di esecuzione è valutata alla fine del corpo del ciclo.

[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. (4 punti)

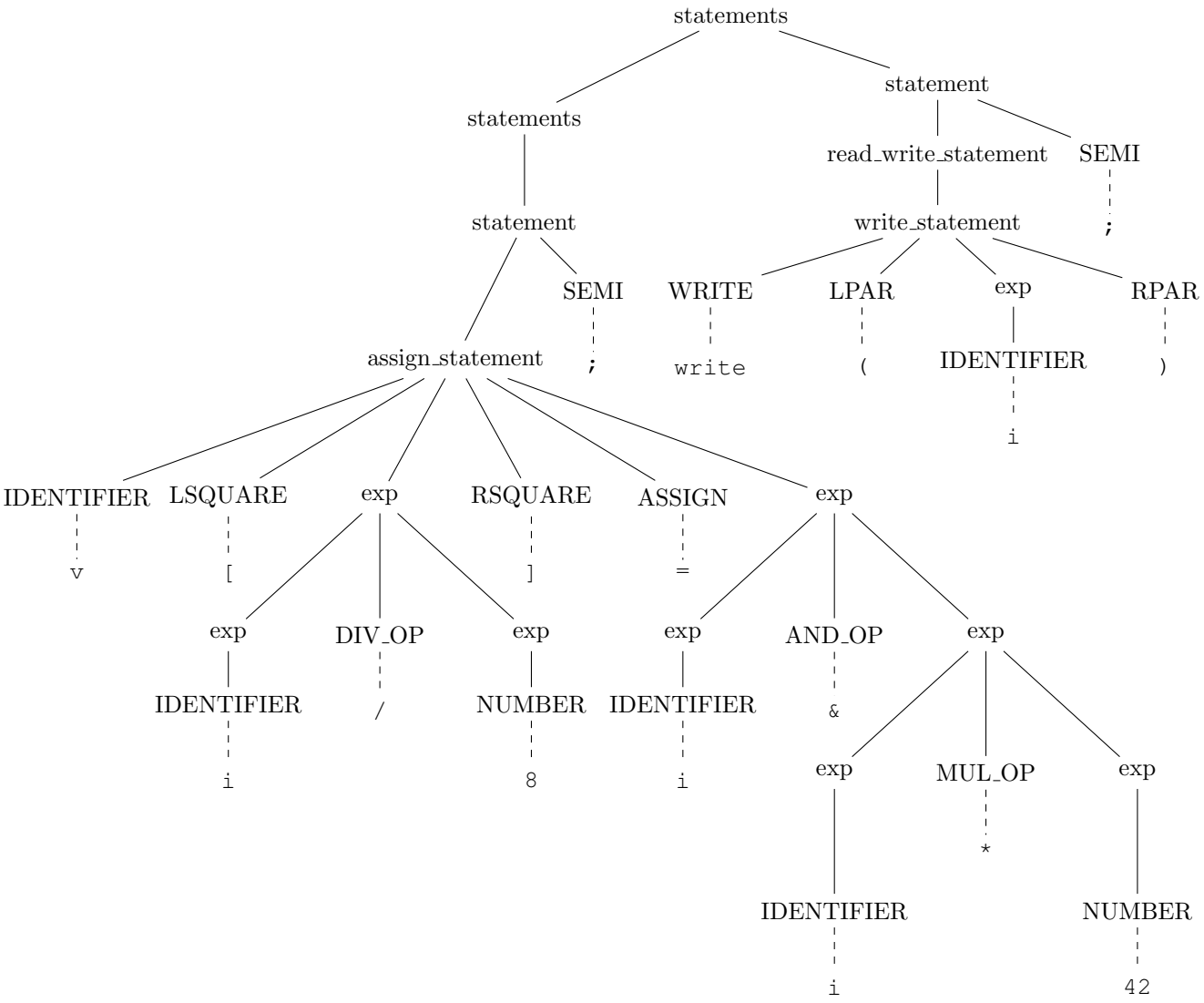
3. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (18 punti)

   La soluzione è riportata nella patch allegata.

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

```
int v[100], i;
v[i / 8] = i & i * 42;
write(i);
```

Scrivere l'albero sintattico relativo partendo dalla grammatica Bison definita in `Acse.y` *iniziando dal non-terminale* `statements`. (6 punti)



5. (**Bonus**) Descrivere come estendere il costrutto *loop-decreasing* affinchè usi una espressione come valore del contatore anziché una variabile.
   Riportare la grammatica `bison` modifica che mostri *entrambe le varianti* del costrutto *loop-decreasing*.
   Un'implementazione completa è opzionale.

```
int x,y,i;
read(x);
read(y);
loop_decreasing
  from x + 100 by 14 - y {
  i = i + 1;
  write(i);
} while (y + i < 100);
```