Linguaggi Formali e Compilatori
Proff. Breveglieri, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
05/03/2014

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 i *costrutti* `map` e `reduce`. Si consideri il seguente snippet di codice come **esempio** dell'utilizzo dei costrutti `map` e `reduce`:

```
int vett[100];
int elem, t, sum;

map elem on vett as {
  t = elem * elem;
  t = t + 2 * elem;
  elem = t - 9;
}

read(t);
sum = 0;

reduce elem into sum
  as [[ sum + t * elem ]]
  on vett;

write(sum);
```

Il costrutto `map` è utilizzato per applicare una trasformazione inplace degli elementi di un array (es. `vett`). Per ogni elemento (es. `elem`) dell'array, viene eseguito il blocco di istruzioni rappresentante la trasformazione e alla fine l'elemento viene riscritto nell'array.
Il costrutto `reduce` è utilizzato per applicare una funzione di riduzione a scalare degli elementi di un array (es. `vett`). Il risultato della riduzione (es. `sum`) viene aggiornato

[1] Tempo 60'. Libri e appunti personali possono essere consultati.
È consentito scrivere a matita. Scrivere il proprio nome sugli eventuali fogli aggiuntivi.

per ogni elemento (es. `elem`) dell'array con il risultato dell'**espressione** di riduzione racchiusa tra doppie parentesi quadre.

```
int elem;
int vett[3];

// vett = {2, -10, 9}

map elem on vett as
  elem = elem * 2;

// now vett = {4, -20, 18}
```

Figura 1: Using `map` to double the elements of an array.

In figura 1 è mostrato una semplice utilizzo di `map` il cui effetto è di raddoppiare il valore di ogni elemento `elem` dell'array `vett`.

```
int elem, sum = 0;
int vett[3];

// vett = {2, 6, 13}

reduce elem into sum as
  [[ sum + elem ]] on vett;

// sum = 21
write(sum);
```

Figura 2: Using `reduce` to compute the sum of the elements of an array.

In figura 2 è mostrato un semplice utilizzo di `reduce` il cui effetto è di accumulare in `sum` i valori dell'array `vett`.

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 implementare il costrutto `map` . (10 punti)

4. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per implementare il costrutto `reduce`. (15 punti)

   La soluzione è riportata nella patch allegata.

5. (**Bonus**) Si descriva come estendere il costrutto `reduce` in modo da ammettere anche una variante con `map` integrato come mostrato dal seguente esempio:

```
int vett[100];
int elem, t,  sum = 0;
reduce elem into sum
  as [[ sum + elem ]]
  on map vett as {
    t = elem * elem;
    elem = 2 + elem;
  }
```