Linguaggi Formali e Compilatori
Proff. Breveglieri, Crespi Reghizzi, Morzenti
Prova scritta [1]: Domanda relativa alle esercitazioni
17/09/2014

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 l'istruzione **cond**. La seguente Figura 1 ne riporta un esempio. L'istruzione

```
int x, y;

read(x);

cond{
  case x>0: read(y);
  case x+1: x=0; y=0;
  default: y=x;
}

write(x);
write(y);
```

Figura 1: Esempio

cond raggruppa una lista, non vuota, di case condizionali eventualmente conclusa con un caso **default**. Ogni case condizionale è identificato dal token **case** seguito da un'espressione, dal simbolo : e da una lista, non vuota, di istruzioni. Un case condizionale viene eseguito se l'espressione associata è verificata (ossia, diversa da zero) ed esso è il primo case condizionale, della lista di case nel corpo del costrutto, la cui espressione sia verificata (*meet-first*). In altre parole, questo accade quando tutte le espressioni dei case condizionali precedenti al case eseguito, non sono verificate. Il caso di default, qualora definito, è eseguito se nessuno dei case condizionali precedenti viene eseguito. NOTA

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

BENE: Al termine dell'esecuzione di un case condizionale, il flusso d'esecuzione passa all'istruzione successiva a **cond**.

Ad esempio, se $x = 1$ dopo l'istruzione $read(x)$, successivamente viene eseguita l'istruzione $read(y)$ e poi le due istruzioni $write$. Se $x = 0$ dopo l'istruzione $read(x)$, viene eseguita la sequenza $x = 0; y = 0;$ e poi le due istruzioni $write$. Se $x = -1$ dopo l'istruzione $read(x)$, viene eseguito il caso **default**.

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

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

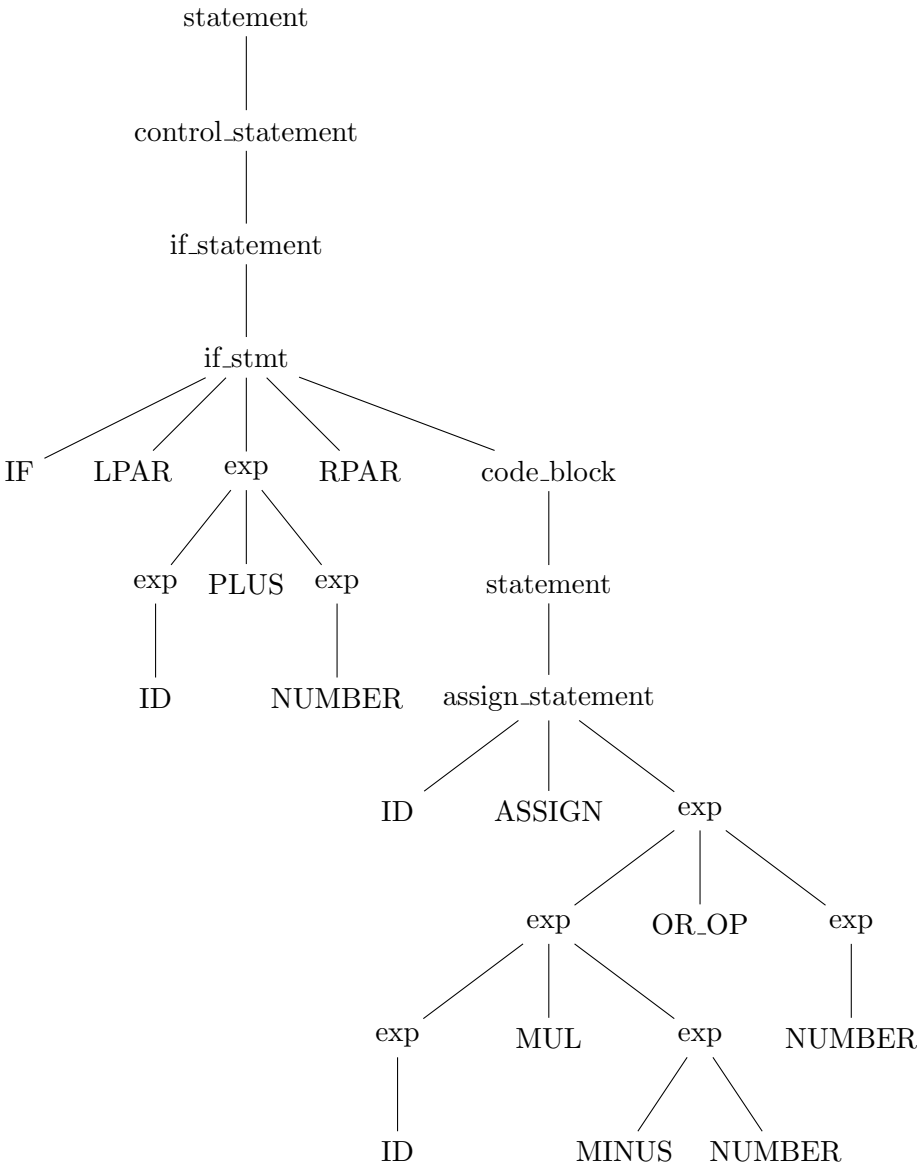
3. Definire le azioni semantiche (o le modifiche a quelle esistenti) necessarie per ottenere la funzionalità richiesta. (18 punti)

   La soluzione è riportata nella patch allegata.

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

```
if(x+1)
  x=y*-2|1;
```

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



5. (**Bonus**) Spiegare come estendere il costrutto **cond** definito nel punti precedenti affinchè si possano definire case condizionali in cui la lista di istruzioni associata sia vuota. In questo caso, la semantica prevede di considerare le espressioni dei **case** come se fossero scritte mediante disgiunzione logica (in altre parole, in OR tra loro). Ad esempio, in Figura 2, le istruzioni $x = 0; y = 0;$ sono eseguite se $x > 0 \vee x + 1 \neq 0$.

```
int x, y;

read(x);

cond{
  case x>0:
  case x+1: x=0; y=0;
  default: y=x;
}

write(x);
write(y);
```

Figura 2: Esempio

La variante proposta richiede una nuova regola sintattica che definisca i case condizionali a lista vuota. L'azione semantica associata aggiunge un'istruzione branch (**bne**) dopo l'espressione exp del case condizionale (a lista vuota) corrente che porti il flusso del programma allo statement definito all'interno del primo case condizionale, a lista non vuota, successivo. L'etichetta di salto viene mantenuta all'interno della struttura associata all'istruzione **cond** attualmente in fase di parsing.