Segunda Prova de Construção de Compiladores.
Primeiro Semestre de 2001. Dept. de Computação – UFSCar.
José de Oliveira Guimarães.
Turma B (Quinta).

Um homem que tem o conhecimento mas sem a capacidade de expressá-lo claramente não é melhor do que aquele que não possui conhecimento de modo algum

Tulcídides

Um cientista da computação é igualmente um cientista e um escritor — esforce-se para aprender a outra metade de sua profissão.

Richard Gabriel

Lembre-se: justifique tudo a menos de menção em contrário.

1. (2.0) Seja a gramática

P ::= VarDec Assigns | Assigns
VarDec ::= "var" ID IDList ";"
IDList ::= | "," ID IDList
Assigns ::= Assigns Assign | Assign
Assign ::= ID "=" N

Onde N e ID são terminais. Calcule a função first para cada não terminal. Lembre-se: first(X) = { X } para X terminal, se X ::= ϵ for uma produção, adicione ϵ a first(X) e se X for não terminal e X ::= $Y_1 Y_2 ...$ Y_k for uma produção, então coloque f em `first`(X) se, para algum i, f $\in$ `first`(Y_i) e $\epsilon \in$ `first`(Y_j), $j = 1, 2, ... i-1$. Se $\epsilon \in$ `first`(Y_j), $j = 1, 2, ... k$, então coloque ϵ em `first`(X).

Não é necessário justificar, naturalmente.

2. (2.0) Fatore à esquerda e elimine a recursão à esquerda da seguinte gramática:

E ::= E E (+|-|*|/)
E ::= Numero

Não é necessário justificar explicitamente, mas deixe claro os passos que você seguiu até a gramática resultante.

3. (3.0) Faça o autômato finito que reconheçe a seguinte expressão regular. Faça ambos o desenho e o código em Java.

```
[0-9]+(E|e)[a-z]*
```

Não é necessário justificar.

4. (3.0) Pode ser provado que se uma gramática G é LL(1) então se A ::= α | β forem duas produções distintas de G, não há um terminal b tal que b $\in$ first(α) e b $\in$ first(β).

Explique porque esta regra é necessária para que G não seja ambígua. Para auxiliá-lo na resposta, encontre uma sentença que possua duas derivações à esquerda diferentes se esta regra for violada.

5. (3.0) Dada a gramática

1. E ::= T E′ 2. E′ ::= + T E′ 3. E′ ::= ϵ
4. T ::= F T′ 5. T′ ::= * F T′ 6. T′ ::= ϵ
7. F ::= (E) 8. F ::= id

onde +, *, (,) e id são terminais, a tabela associada, que chamaremos de `M`, é

	id	(	)	+	*	eof
E	1	1	–	–	–	–
E′	–	–	3	2	–	3
T	4	4	–	–	–	–
T′	–	–	6	6	5	6
F	8	7	–	–	–	–

O símbolo – deve ser lido 0 (zero).

Faça a análise da sentença `id + id`. Utilize uma tabela como a abaixo. A ação pode ser "emp n" para utilizar a regra n para substituição, "desemp" para reconhecer o token corrente (avança-se para o próximo token) e "aceita" para terminar a análise.

Pilha	Entrada	Ação
E	id + id	

Não é necessário justificar.