Trabalhos Compiladores (individual ou 2 alunos)

2º bimestre: (prova 60%, trabalhos 40%)

Especificação da gramática da linguagem (40%):

- Especificar qual o tipo de analisador sintático será implementado:
    - analisador descendente preditivo recursivo;
    - analisador descendente preditivo tabular;
    - analisador de precedência de operadores;
    - analisador ascendente SLR(1);
    - analisador misto (especificar onde cada tipo será usado).

- A gramática deverá ser especificada de acordo com o que foi visto em sala de aula, **especificando precedência e associatividade para os operadores**.

- A especificação das produções devem seguir a simplificação do padrão EBNF ISO/IEC 14977: 1996(E) como usada em sala de aula (nomes sem espaços e sem vírgulas entre símbolos nas produções).

- **Não** são admitidas formas **condicionais** ou com **repetições** nas produções, já que as mesmas não são tratáveis pelos analisadores estudados.

- Dependendo do analisador sintático escolhido deverão ser especificados adicionalmente:
    - analisador descendente preditivo recursivo ou tabular:
        - a especificação LL(1) correspondente da gramática, com as restrições necessárias;
    - analisador descendente preditivo tabular, analisador de precedência de operadores ou ascendente SLR(1):
        - tabela de análise, como vistas em sala de aula;
    - analisadores mistos:
        - cada parte de acordo com o analisador utilizado.

Analisador sintático(60%):

- Deverá ser implementado o analisador especificado no item anterior.

- Deverão ser incluídas ações semânticas que permitam desenhar a árvore de derivação, a cada derivação/redução, imprimir a produção usada, bem como identificar os tokens identificados/empilhados. Cada informação deverá estar em uma linha separada, sendo que os tokens devem ser emitidos indentados com pelo menos 4 espaços em branco de acordo com o formato:

```
    token = [categ, lexema, linha, col]
```

Exemplo:

```
aa + bb * cc
1...5...10..
enum categ {Opa, Opm, Id};
```

- descendente: (tokens são listados quando identificados)

Gramática:	Resultado esperado:

`Ea  = Ta Ear`
`Ear = 'opa' Ta Ear`
`Ear = ϵ`
`Ta  = Fa Tar`
`Tar = 'opm' Fa Tar`
`Tar = ϵ`
`Fa  = 'id'`
	`Ea = Ta Ear`
	`Ta = Fa`
	`Fa = 'id'`
	`  id = [2, aa, 001, 01]`
	`Ear = 'opa' Ta Ear`
	`  opa = [0, +, 001, 04]`
	`Ta = Fa Tar`
	`Fa = 'id'`
	`  id = [2, bb, 001, 06]`
	`Tar = 'opm' Fa Tar`
	`  opm = [1, *, 001, 09]`
	`Fa = 'id'`
	`  id = [2, cc, 001, 11]`
	`Tar = epsilon`
	`Ear = epsilon`

- ascendente (SLR): (tokens são listados quando empilhados)

Gramática:	Resultado esperado:

`Ea = Ea 'opa' Ta`
`Ea = Ta`
`Ta = Ta 'opm' Fa`
`Ta = Fa`
`Fa = 'id'`
	`  id = [2, aa, 001, 01]`
	`Fa = 'id'`
	`Ta = Fa`
	`Ea = Ta`
	`  opa = [0, +, 001, 04]`
	`  id = [2, bb, 001, 06]`
	`Fa = 'id'`
	`Ta = Fa`
	`  opm = [1, *, 001, 09]`
	`  id = [2, cc, 001, 11]`
	`Fa = 'id'`
	`Ta = Ta 'opm' Fa`
	`Ea = Ea 'opa' Ta`

Os exemplos foram gerados virtualmente e podem conter algum equívoco.

-
  - no caso do analisador de precedência de operadores ou de analisador misto, adequar de acordo com os modelos acima, em caso de dúvida consultar o professor.

- Os exemplos dos trabalhos do 1º bimestre deverão ser analisados.

SUGERE-SE FORTEMENTE que versões prévias sejam apresentadas ao professor para análise e discussão.