Prova 1 (2016.2)

1 - Tinha um trecho de código, e era pra dizer os erros léxicos que tinham nele, e também os tokens que iriam ser gerados(na reposição). **(Trecho de código Java, uma classe Pessoa com um método, identificar os tokens que iriam ser gerados).**

2 - Gerar uma expressão regular para gerar linguagem de números binários múltiplos de 4. Alfabeto era {0,1}. **R: [0|1]*00**

3 - Transformar uma expressão regular em um autômato finito não determinístico (NDA) usando o algoritmo de Yamada.
4 - Transformar a mesma expressão regular da questão 3 em um autômato finito determinístico **DIRETAMENTE**, ou seja, usando árvore sintática e as funções nullable, firstpos, lastpos e followpos, mostrando cada passo.

Prova 2 (2016.2)

1 - Elimine a recursão à esquerda da gramática mostrando cada passo.
2 - Dada a gramática G:

- Faça os FIRST(x)
- Faça os FOLLOW(A)
- Construa a tabela de análise sintática descendente

3 - Usando a gramática da questão 2, resolva as funções CLOSURE(I) e GOTO(I, simbolo).

4 - O que é tratamento de erro por produção de erro? Dê um exemplo na gramática da questão 2

Prova 3 (2016.2)

1 - Construa uma DDS para (do/while if)

Reposição: S1;
If B1 then {S2, S3}
else do S4
While alguma coisa
S5; // mais ou menos isso

2 - Construir uma DDS completa para

1. Uma expressão específica de java que ele descrevia (Integer, Double, Boolean)

Reposição: a) DDS para casting de expressão, eu entendi que fosse uma DDS pra algo assim: S -> var = (type) Exp
Ai só o boolean n permitia o casting

b) Acho q pra esse era algo do tipo:
S -> var = metodo

Ele dizia que tinha que checar o nome, atributos, tipo de retorno do método e tal...

2. Uma expressão lógica que ele descrevia (aqui só envolvia tributos true false, AND, OR, NOT, id)

3 - Fazer o código de 3 endereços para um trecho de código que ele dava (envolvia while e if/else -> parecido com: if (x < 100 || (x > 200 && x != y))

Reposição: Tinha while com if/else tb, algo do tipo:
While d[z][z] < 10 and x > 5
if d[z][z] = 0 ...

Prova 4 (2016.2)

1 - Gere o código objeto para os seguintes comandos de três endereços, considerando a alocação de pilha onde o registrador SP aponta para o topo da pilha e que os tamanhos dos registros de ativação para os procedimentos m1 e m2 e tenham sido determinados como m1size e m2size, respectivamente.:

```
//codigo de m1
Int a = 2;
Do {
Call m2;
A = a+1
} while (a < 5)
Halt


//código de m2
Int b = 2;
Int c = b;
If (b != 0 && x[b] > não_lembro) {
Faz umas coisas envolvendo array
Ex: x[b] = c; d = x[c], etc
}
Faz outras operações
```

return

2 - Construa o DAG para bloco básico abaixo e o otimize (simplifique), considerando que apenas a e d estão vivos no bloco. Explique a otimização passo-a-passo.

Não lembro exatamente, mas começava normal, e no final tinha umas operações com array
ex:
d = b * c
e = a + b
b = b * c
a = e + d
F[b] = c
A = f[c]

3 - Feito em sala, construir uma arquitetura MDA (aquela das caixinhas) para transformar Pascal em Assembly, usando 3 endereços no meio para passar pra DAG e otimizar, depois volta para três endereços, depois Assembly e gera o código.

PROVA FINAL 2016.2

1 - Código java com classe pessoa, apontar a) erros léxicos, b) erros sintáticos, c) erros semânticos. **Tinha 1 de cada erro, bem fácil. (2.0)**

2 - Tirar a recursão à esquerda de uma gramática. **A gramática era bem simples e a recursão estava bem definida, só usar a regra que tem nos slides. (2.0)**

3 - Dizer quais os erros possíveis num analisador SLR e explicar o que são e como acontecem. **Resposta é aquele negócio de shift-reduce e reduce-reduce. Tem que explicar. (1.0)**

4 - Fazer DDS de uma especificação que ele deu, atribuição de variável usando operadores lógicos (and,not, or) e valores eram (true,false,identificador). **(2.0)**

5 - Explicar um método de otimização de código e ilustrar (**ex: eliminação de código morto e usar uma DAG para mostrar) (1.0)**

6 - Gerar o código assembly para um código. **Muito parecido com o do slide. Tem que usar a pilha para guardar os endereços de volta, dar os branches, etc. A chamada de "m2" era dentro de um while. MUITO parecido com a primeira questão a prova 4. (2.0)**