**Vamos criar a lista de possíveis questões da prova final...**

**Vou colocar abaixo as questões que me lembro que cairam na prova e jogar lá pro fim do documentos as questões da outra prova**

Primeiro estágio

Qual a diferença entre compiladores convencionais Java e Compiladores JITs Java? Explique do ponto de vista arquitetural de um compilador.

Escreva uma definição regular pra uma linguagem que contêm exatamente cinco vogais diferentes e em ordem, além das consoantes, que podem ocorrer em qualquer lugar, repetidamente ou não, e em qualquer ordem.

Converta a expressão (a|b)*aa(a|b)* para um NFA, usando o algoritmo Yamada-e-naum-sei-mais-o que.

Converta a expressão regular (a|b)*aa(a|b)* diretamente em um DFA usando as funções nullable, firstpos, lastpos e followpos.

Segundo Estágio

A gramática abaixo é adequada para a análise sintática descendente. Se não transforme para ser.

bexp -> bexp or bterm | bterm
bterm -> bterm and bfactor | bfactor
bfactor -> not bfcator | (bexp) | true | false

Usando a gramática resultante G' gerada pela questão anterior:
a) Calcule FIRST(X).
b) Calcule FOLLOW(A).
c) Construa a tabela de análise para o reconhecedor sintática preditivo de G'.

Sobre Analisadores Sintáticos SLR e tables SLR:
a) Defina o detalhe em pseudo-código os algoritmos para se calcular os valores

das funções closure e goto, presentes em um analisador sintático SLR e essenciais para a construção de tabelas SLR.

Aplique a função closure para uma coleção contendo um item de uma gramática estendida definida sobre a gramática G' resultando da questão 1.

Quarto  Estágio

Construa o código de três endereços para o comando abaixo, com o arranjo com 4 bytes:

```
for ( i = 0; i < 5; i++) {
  if x[i] < 25 then x[i] = 0;
  else {
          x[i] = 1;
          y = x[i + 1]
  }
}
```

construa o grafo de fluxo e diga aonde tem um loop

Construa DAG considerando a vivo

```
d = g * c
f = a + g
b = g * c
a = f - d
```

Gere código objeto para o código de 3 endereços abaixo,

```
//m
call p
call q
halt
//p
action1
return
```

```
//q
action2
call p
call q
call p
return
```

Em que fases do processo de compilação a transformações de modelos e transformações textuais pode ser usada e como podem ser usadas?

Podem ser utilizadas na geração de código intermediário ou na otimização de código.
Otimização: modelo de origem = modelo de destino, templates de otimização definem as transformações.
Geração de Código Intermediário: modelo de origem não é necessariamente = ao modelo de destino, pode ser de PSM pra código...[E a transformação de modelo para texto]

## MDA e Compiladores

Quais os possíveis potenciais do uso de MDA e compiladores?
MDA possui potencial para gerar código intermediário ou código objeto, é possível especificar transformações a partir do meta-modelo para gerar estes artefatos.

MDA ainda possui potencial para refatoramentos, onde a linguagem fonte e a linguagem destino são a mesma, assim como realizar otimizações no código.

## Como a análise semântica pode ser feita através de meta-modelos, exemplifique

A análise semântica é feita em meta-modelos a partir de regras OCL.Por exemplo, em um metamodelo de banco de dados, se é gerada uma regra para criar um script de inserção, uma regra ocl pode definir em uma pré condição que a chave primária não deve existir na tabela ou ainda que o número de tuplas da inserção deve ser igual ao número de tabelas, sei lá, é só um exemplo doido.

## Como é feita a análise léxica com meta-modelos ??

## Quais as técnicas de transformação de modelos?

Baseada em templates
Baseada em Modelos
Baseada em Programação

## ATL

Um módulo ATL é composto por Header, Import, Helpers e Rules
Header: apresenta os identificadores da linguagem de entrada e saída
Import: Utilização de bilbiotecas auxiliares
Helpers: Tipo de função auxiliar (não entendi bem)
Rules: Regras de transformação propriamente ditas

## Diferencie Matched Rules de Called Rules

*Matched Rules* são declarativas, não precisam ser explicitamente invocadas. São baseadas em padrões. Basicamente especificam como vai ser a transformação do modelo de entrada pra o de saída.
*Called Rules* são imperativas, devem ser invocadas, semelhante aos helpers.

## Cite X características desejáveis em uma linguagem de transformação:

Linguagem de consulta

Linguagem para definição de transformações
Sintaxe abstrata
Sintaxe concreta
Ser declarativo
Ser imperativo
Operar sobre modelos

Questão do período passado! Vamos responder!!!!

1. quais módulos de MDA que determinam os nós terminais e os nós não-terminais?

primeiro: quais sao os módulos de MDA?? kkkkkkk

Acho que o correto não é módulos e sim modelos, se forem modelos, teríamos o PIM, CIM, PSM e o Código...

Nós terminais poderia ser PSM e Código, visto que eles dependem da plataforma e tecnologias utilizadas suas transformações indicariam o fim de uma derivação.
Já PIM e CIM seriam não-terminais, suas derivações produziriam outros modelos, que por sua vez poderiam ser transformados em módelos de mais baixo nível, até chegar no código...
Faz sentido isso..

2. Comente sobre as questoes dos modelos convecionais de compiladores que não é tratado pelos compiladores MDA
seria isso?
- Análise léxica (como vai separar o modelo em tokens?)
– Qual algoritmo de análise sintática a ser empregado (é empregado algum algoritmo de analise sintatica? eu vi la q a compilação ainda é manual.. faz sentido isso?????????)

análise sintática: como entender o relacionamento entre os artefatos do modelo? é possível especificar uma ordem? sem uma ordem clara e bem estabelecida é impossível identificar se o modelo casa com o meta-modelo do mesmo

– Recursões e fatoramentos em gramáticas?

recursões caí no mesmo problema acima, não tem como estabeler uma ordem em um meta-modelo, daí não tem como identificar uma recursão a direita ou esquerda, você identifica a recursão, porém não tem como tratá-la.

– Onde se encaixam as transformações textuais?

É necessário existir uma transformação do modelo para texto para que a geração automática

de artefatos seja feita através do modelo, desde documentação até código executável, como exemplo podemos citar a linguagem de geração textual mofscript

3. Outra perguntava quais artefatos na arquitetura de compiladores MDA!
UML/OCL e Perfis
??

Artefatos de um compilador MDA:
Inicialmente é necessário ter dois meta-modelos, um declarando os elemtnos da linguagem fonte e outro para a linguagem destino. Depois disso é necessário ter algum engine ou linguagem de transformação entre estes modelos. Se necessário (depende do nível de abstração do compilador) tem que ter uma linguagem de geração textual como MOFScript. Eu acho que é isso. (bracinho)

4. Cite as vantagens e desvantagens das seguinte transformações MDA..

- Baseada em programação
    -Vantagens possui frameworks que auxiliam na codificação das transformações
    -Desvantagens limitado a linguagens específicas, caso os requisitos mudem, ou alguma novo elemento seja adicionado ao modelo é necessário adicionar esta transformação ao código.
- Baseada em templates
    -Vantagens: é parametrizável, parâmetros são substituídos por informações no modelo destino. Declarativo.
    -Desvantagens: nem sempre uma regra pode casar com um template, podem existir gaps (resposta avulsa)
- Baseada em modelos
    -Vantagens: O processo de criação do código também gera material de documentação do programa, possibilidade de utilizar o mesmo modelo para gerar código em várias linguagens.
    -Desvantagens: nem sempre é eficiente, existem diversos padrões: Ausência de um padrão e uma semântica formal bem definida.

Pessoal: Slide 45 de transformações de modelo tem uma tabelinha, com uns dados massas, tipo concisão, eficiência, expressividade, etc.

**AMANHÃ TODO MUNDO VOLTANDO DE BIKE PRA CASA!!!!!!!!!!**
**?**