

# Teste 2 – 2021

## Informação

Considere o seguinte excerto de um programa de Programação em Lógica com Restrições (PLR):

```
: - use_module(library(clpfd)).  
:- use_module(library(lists)).  
  
restricao_produto([], _, 1).  
restricao_produto([Pos|Ps], Nums, Prod):-  
    nth1(Pos, Nums, Num),  
    Prod #= Prod1 * Num,  
    restricao_produto(Ps, Nums, Prod1).  
  
fatores(Nums, Prod, NPosicoes, NPares, Posicoes):-  
    length(Posicoes, NPosicoes),  
    all_distinct(Posicoes),  
  
    restricao_produto(Posicoes, Nums, Prod),  
  
    % Predicados a implementar nas perguntas 3 e 4  
    restricao_pares(Posicoes, Nums, NPares),  
    restricao_remover_simetrias(Posicoes),  
  
    labeling([], Posicoes).
```

O predicado principal, **fatores(+Nums, +Prod, +NPosicoes, ?NPares, -Posicoes)**, seleciona **NPosicoes** números da lista de inteiros **Nums** cujo produto é igual a **Prod**. **Posicoes** indica os índices da lista **Nums** que foram selecionados (com os índices a começar em 1). **NPares** representa a quantidade de números pares selecionados (a ser implementado na pergunta 3).

## Pergunta 1

Correta Pontuou 1,000 de 1,000

O excerto de código apresentado acima tem UM e exatamente UM erro.

Em qual das fases da definição do problema de PLR é que o erro está presente?

Selecione uma opção de resposta:

- a. Definição do domínio das variáveis de decisão
- b. Definição da pesquisa no espaço de estados (labeling)
- c. Definição do número de variáveis de decisão
- d. Definição das restrições

## Pergunta 2

Respondida Pontuou 1,000 de 1,000

Copie e modifique o excerto de código do enunciado de forma a corrigir o erro que identificou.

**Nota:** Apenas deve adicionar, remover ou modificar UMA linha de código. Deixe o resto do código como apresentado.

## Pergunta 3

Respondida Pontuou 2,500 de 2,500

Implemente o predicado **restricao\_pares(+Posicoes, +Nums, ?NPares)**, invocado pelo excerto de código de códido acima.

Este predicado define restrições de forma a assegurar que a quantidade de inteiros pares selecionados em **Nums** é igual a **NPares**.

**Dica:** A definição deste predicado requer apenas cerca de 6 linhas de código.

## Pergunta 4

Respondida Pontuou 2,500 de 2,500

Implemente o predicado **restricao\_remover\_simetrias(+Posicoes)**, invocado no excerto de código.

Este predicado define restrições de forma a impedir que sejam geradas soluções redundantes.

Por exemplo, a solução **Posicoes = [2,3,1]** é redundante se já foi dada a solução **Posicoes = [1,3,2]**, pois ambas indicam que foram selecionados os primeiros três inteiros da lista **Nums**.

**Dica:** A definição deste predicado requer apenas cerca de 4-5 linhas de código

## Pergunta 5

Respondida Pontuou 4,000 de 4,000

O Sr. Asdrúbal Amaral Silvino organiza todos os anos uma grande festa de Passagem de Ano. Para o evento, como dita a tradição, reserva um piso inteiro do seu restaurante preferido, Faisão Para Onze (FPRO).

O Sr. Amaral Silvino sabe quanto custa cada prato da ementa e qual o total que foi pago, mas não sabe quanto é que foi pedido de cada prato. No entanto, sabe quanto é que foi consumido do prato mais pedido e tem conhecimento de uma particularidade: todos pratos que foram consumidos na festa tiveram um número de pedidos distinto. Poderá haver um ou mais pratos da ementa sem pedidos.

Implemente o predicado **pratos(+PrecosUnitarios, +Total, +QuantidadeMaxima, -Quantidades)** que determina quantos pedidos é que houve de cada prato durante a festa. **PrecosUnitarios** é uma lista com os preços unitários (números inteiros) de cada prato. **Total** é um inteiro com o preço total da comida durante a festa. **QuantidadeMaxima** é a quantidade pedida do(s) prato(s) com mais pedidos.

**Quantidades** é uma lista de inteiros que indica quantos pedidos houve de cada prato. O i-ésimo elemento corresponde ao número de pedido para o i-ésimo prato de **PrecosUnitarios**.

Para a instância do problema abaixo, há duas soluções possíveis. A primeira solução indica que houve 6 pedidos do primeiro prato (que custa 10) e 9 do quarto prato (que custa 8). O segundo e o terceiro prato não tiveram pedidos. **[2,1,2,9]** não é uma solução deste problema pois, dos pratos consumidos, haveria dois com o mesmo número de pedidos (o primeiro e o terceiro).

```
| ?- pratos([10,22,9,8], 132, 9, Quantidades).
Quantidades = [6,0,0,9] ? ;
Quantidades = [9,0,2,3] ? ;
no
```

**Dica:** Uma definição compacta deste predicado requer apenas cerca de 8 linhas de código.

## Pergunta 6

Respondida Pontuou 4,900 de 5,000

A Dona Conceição tem um negócio de autocarros, Autocarros Escolares e Desportivos da Amadora (AEDA), que todos os anos leva famílias e grupos de amigos de todos os pontos do país a assistir ao Festival do Chocolate em Óbidos. A empresária tem uma frota de vários autocarros, cada um com uma capacidade máxima para passageiros específica. Os grupos que usam o serviço de transporte da Dona Conceição têm diversas dimensões.

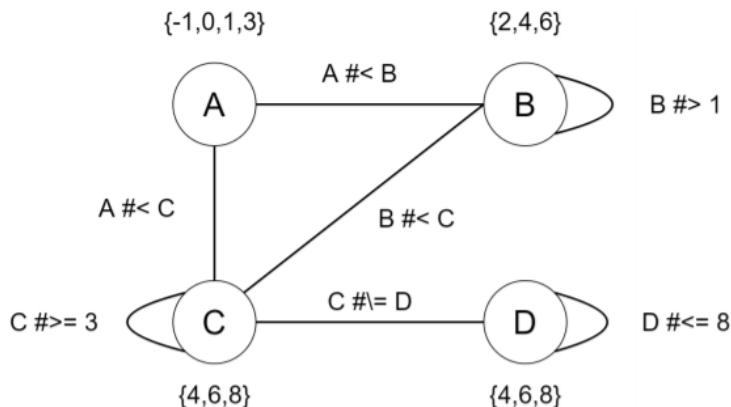
Implemente o predicado **autocarros(+Grupos, +Autocarros, -IDsAutocarros, -NAutocarrosUsados)** que determina em **IDsAutocarros** qual o identificador do autocarro a usar por cada grupo de pessoas, de forma a minimizar a quantidade de autocarros usados. Não devem ser feitas alocações de pessoas a autocarros de forma a que pessoas do mesmo grupo estejam em autocarros diferentes, ou seja, cada grupo de pessoas não pode ser decomposto. Se houver algum grupo com tamanho superior à capacidade dos autocarros, não existe solução para o problema. **Grupos** é uma lista de inteiros que representa o número de elementos de cada grupo de pessoas. **Autocarros** é uma lista de inteiros com a capacidade máxima de cada autocarro. **IDsAutocarros** é uma lista de mesmo comprimento que **Grupos** no qual o i-ésimo indica qual o identificador (ID) do autocarro a ser usado pela i-ésimo grupo em **Grupos**. O ID do autocarro é a sua posição na lista **Autocarros** (com os índices a começar em 1). **NAutocarrosUsados** é um inteiro com o número total de autocarros usados para transportar todos os grupos em **Grupos**.

```
| ?- autocarros([3,5,7,4,3], [11,14,10,20], IDs, N).  
IDs = [1,1,2,2,1],  
N = 2 ?
```

## Pergunta 7

Incorreta Pontuou -0,200 de 1,000

Considere o seguinte grafo de restrições, representativo de um problema de satisfação de restrições:



Este grafo é:

Selecione uma opção de resposta:

- a. 1-consistente, 2-consistente e 3-consistente
- b. 1-consistente e 2-consistente, mas não 3-consistente
- c. 1-consistente, mas não 2-consistente ou 3-consistente
- d. Nem 1-consistente, nem 2-consistente, nem 3-consistente
- e. 1-consistente e 3-consistente, mas não 2-consistente

## Pergunta 8

Correta Pontuou 1,000 de 1,000

Considere novamente o grafo de restrições da pergunta anterior. Assuma que o grafo representa o estado de um problema antes da chamada ao labeling, e que a chamada é feita com os seguintes parâmetros:

```
labeling([ff, step, down], [A,B,C,D])
```

Qual é a primeira etiqueta (label) parcial a ser gerada pelo SICStus?

Selecione uma opção de resposta:

- a. {C - 8}
- b. {A - 3}
- c. {C - 4}
- d. {D - 8}
- e. {B - 6}

## Pergunta 9

Correta Pontuou 1,000 de 1,000

Qual das seguintes ações não é possível de efetuar usando a biblioteca clpfd do SICStus Prolog?

Selecione uma opção de resposta:

- a. Definir que a escolha da próxima variável a etiquetar (label) é aleatória.
- b. Definir duas estratégias de pesquisa distintas para cada metade das variáveis de decisão.
- c. Definir um tempo máximo para a fase de pesquisa.
- d. Definir que a escolha do valor da próxima variável a etiquetar (label) é aleatória.
- e. Definir o número de variáveis de decisão como uma variável de domínio.

## Pergunta 10

Incorreta Pontuou -0,200 de 1,000

Qual das seguintes afirmações está correta?

Selecione uma opção de resposta:

- a. O paradigma Constraint & Generate é superior ao Generate & Test, tendo uma performance temporal igual ou superior ao paradigma Generate & Test em qualquer problema computacional.
- b. Todas as restantes afirmações estão erradas.
- c. Se um problema de satisfação de restrições tiver múltiplas soluções, então estamos perante um caso de simetria.
- d. A restrição *all\_distinct/1* é mais restritiva do que a restrição *circuit/1*.
- e. Os sistemas de programação baseados em restrições seguem um paradigma de programação em lógica.