	Linguagem de Programação e Paradigmas *Prof. Esp. Ademar Perfoll Junior* Sistemas de Informação

Trabalho 03 - Programação Lógica

Objetivo: Fixar os conhecimentos práticos referentes à programação lógica.
Trabalho pode ser realizado de forma individual ou em duplas.
O desenvolvimento deve ser realizado na linguagem de programação Prolog.

O projeto será dividido em cinco módulos, cada um focando em uma aplicação específica de um sistema especialista. O objetivo é desenvolver diferentes sistemas que deduzem informações ou fazem recomendações baseadas em fatos e regras estabelecidos.

1. Diagnóstico Médico
Objetivo: Criar um sistema especialista que sugira diagnósticos médicos com base em sintomas informados pelo usuário.

Requisitos:

- Definir uma base de sintomas comuns, como febre, dor de cabeça, tosse, etc.
- Implementar regras de inferência para diagnósticos de doenças simples, como gripe, infecção viral ou enxaqueca.
- Permitir a interação com o usuário, fazendo perguntas sobre os sintomas e deduzindo o diagnóstico mais provável.
- Incluir um alerta ou recomendação para consultar um médico se o sistema não puder identificar uma condição específica.

Exemplo de regra:
doenca(gripe) :- febre, tosse, cansaço.
doenca(infeccao_viral) :- febre, dor_de_garganta, dor_muscular.

Apresentar o print do ambiente executando os algoritmos em seu computador, anexar os testes no classroom que comprove que o aluno ou a dupla executou as consultas do Prolog.

---

2. Análise de Problemas de Veículos
Objetivo: Desenvolver um sistema especialista que ajude a identificar problemas comuns em veículos com base em sintomas ou comportamentos relatados.

Requisitos:

- Definir uma base de problemas comuns, como motor não liga, barulho estranho ao frear, ou vazamento de óleo.
- Implementar regras para sugerir soluções, como verificar a bateria, o sistema de freios ou o nível de óleo.
- Permitir ao usuário descrever o problema do veículo e obter recomendações de diagnóstico ou manutenção.
- Oferecer conselhos de segurança, como "leve o carro a um mecânico" se o problema for grave.

Exemplo de regra:
problema(bateria_fraca) :- motor_nao_liga, luzes_fracas.
problema(freio_desgastado) :- barulho_ao_frear, pedal_freio_macio.

Apresentar o print do ambiente executando os algoritmos em seu computador, anexar os testes no classroom que comprove que o aluno ou a dupla executou as consultas do Prolog.

---

3. Recomendações de Livros

Objetivo: Criar um sistema especialista que recomende livros com base em preferências de gênero e interesses do usuário.

Requisitos:

- Criar uma base de dados de livros, categorizados por gênero, como ficção, ciência, história, ou autoajuda.
- Implementar regras que façam recomendações com base nas respostas do usuário sobre seus interesses.
- Permitir ao usuário selecionar múltiplos gêneros ou tópicos de interesse.
- Oferecer recomendações detalhadas, incluindo o título do livro e uma breve sinopse.

Exemplo de regra:
recomendar('1984') :- genero(ficcao), interesse(politica).
recomendar('Sapiens') :- genero(historia), interesse(ciencia).

Apresentar o print do ambiente executando os algoritmos em seu computador, anexar os testes no classroom que comprove que o aluno ou a dupla executou as consultas do Prolog.

---

4. Recomendação de Treinos de Academia

Objetivo: Desenvolver um sistema especialista que sugira treinos de academia baseados em objetivos, como perder peso, ganhar massa muscular ou melhorar o condicionamento físico.

Requisitos:

- Definir uma lista de treinos e exercícios específicos para diferentes objetivos.
- Implementar regras que associem os objetivos do usuário com planos de treino recomendados.
- Perguntar ao usuário sobre seu objetivo, experiência na academia e disponibilidade de tempo.
- Oferecer recomendações detalhadas de treino, incluindo séries, repetições e instruções para cada exercício.

Exemplo de regra:
treino(hipertrofia) :- objetivo(ganhar_massa), experiencia(intermediario).
treino(perda_de_peso) :- objetivo(emagrecer), disponibilidade(tempo_limitado).

Apresentar o print do ambiente executando os algoritmos em seu computador, anexar os testes no classroom que comprove que o aluno ou a dupla executou as consultas do Prolog.

---

5. Análise de Investigação Criminal
Objetivo: Criar um sistema especialista que auxilie na análise de casos criminais, sugerindo suspeitos ou cenários com base em evidências e testemunhos.

Requisitos:

- Definir uma base de evidências e testemunhos, como impressões digitais, motivo, ou testemunha ocular.
- Implementar regras que relacionem as evidências com possíveis suspeitos ou teorias do crime.
- Permitir ao usuário fornecer informações sobre um caso específico e obter uma análise preliminar.
- Incluir opções para recomendar a investigação de determinados suspeitos ou a verificação de álibis.

Exemplo de regra:
suspeito(jose) :- impressao_digital(lugar_crime), motivo(financeiro).
teoria(crime_passional) :- testemunha(visto_discutindo), relacao(intima).

Apresentar o print do ambiente executando os algoritmos em seu computador, anexar os testes no classroom que comprove que o aluno ou a dupla executou as consultas do Prolog.

Critérios de avaliação:
Trabalho vale máximo de 10 pontos divididos da seguinte forma:

Critério	**Pontuação**
Definição de Regras e Fatos:	2,0

● A relevância e clareza das regras lógicas usadas em cada módulo. ● Se as regras conseguem cobrir uma variedade de cenários e fazem inferências precisas.	
Funcionalidade: ● A precisão com que o sistema realiza inferências ou faz recomendações. ● Se o sistema responde corretamente com base nas informações fornecidas pelo usuário.	3,0
Interação com o Usuário: ● A clareza das perguntas e a facilidade de uso da interface textual. ● Se o sistema conduz o usuário de forma lógica até uma resposta ou diagnóstico.	2,0
Originalidade e Complexidade: ● A complexidade do sistema, incluindo a variedade de regras e cenários cobertos. ● Inovações ou personalizações no tema escolhido, como diagnósticos mais complexos ou recomendações personalizadas. ● Entrega dos prints de execução.	2,0
O projeto deve ser publicado no github, contendo a explicação dos membros do grupo, como instalar, configurar e executar a aplicação. O nome de todos os membros deve aparecer no github marcado com @. O código fonte deve conter os comentários principais explicando a implementação. Adicionar o nome dos desenvolvedores ao apresentar o menu do sistema, por exemplo, no rodapé: Desenvolvidor por: [Nome do(a)' Aluno(a)] Sobre o README: A documentação deve explicar o funcionamento do conversor e o uso de programação funcional. Devem ser incluídos exemplos de entrada e saída, além de instruções sobre como rodar o programa.	1,0
**Itens não atendidos dentro do escopo esperado terão 0,10 pontos descontados	