Tópicos em Inteligência Artificial

Prolog

UFFS
Ciência da Computação

27/06/2017

Prova 1

Nome: _____ Matrícula: _____

Prova 1: [130 pontos / 100]

Instruções:

1. A prova será feita nos computadores do Laboratório. Nada de laptop ou computadores pessoais.
2. Coloque seu nome e matrícula no arquivo de repostas.
3. Leia bem a prova. A leitura e interpretação das questões é responsabilidade do aluno.
4. Caso o computador que você está usando não tenha algum software instalado, instale ou solicite para trocar de computador.
5. A prova é com consulta ao material individual do aluno. Não será permitido empréstimo de material. Materiais permitidos: Livros, apostila, resumo, programas.
6. *Não é permitido o uso de internet.* Alunos pegos usando a internet terão nota Zero.
7. Ao final da prova chame o professor para colocar o arquivo resposta no pen drive.
8. Para obter nota 10,0 (Dez) o aluno precisa de 100 pontos. Os pontos extras são para ajudar os alunos. **No entanto a nota máxima continua sendo 10,0**.

Duração da prova: 2 horas e 10 minutos

Boa prova!

Questões:

1. [25 **pts**] **Recursividade:**
Faça predicados em Prolog para:

(a) [5 **pts**] Testar se um número é um número primo.

Solução:

```
eh_primo(X) :- L is sqrt(X), eh_primo(2,L,X), !.
eh_primo(I,L,_):- I > L, !.
eh_primo(I,_,X):- Y is X mod I, Y = 0, !, fail.
eh_primo(I,L,X):- I1 is I+1, eh_primo(I1,L,X), !.
```

(b) [10 **pts**] Dado um número achar o próximo número primo acima do número dado.

Solução:

```
prox_primo(N,P):- P is N + 1, eh_primo(P), !.
prox_primo(N,P):- N1 is N + 1, prox_primo(N1,P), !.
```

(c) [10 **pts**] Dado um valor n achar o n-ésimo número primo. **Observação:** Assuma que o primeiro número primo é 1.

Solução:

```
nesimo_primo(1,1).
nesimo_primo(N,P):- N1 is N-1, nesimo_primo(N1,P1), prox_primo(P1,P), !.
```

2. [20 **pts**] **Listas:**

(a) [20 **pts**] Faça um predicado em Prolog que recebe uma lista com pares de nomes e datas de nascimento, conforme exemplo abaixo, e imprime os nomes por ordem decrescente de idade. **Dica:** Faça um predicado que arrume os dados de entrada num formato mais fácil de ordenar pela data.
Exemplo: [(*joao*,13/07/85), (*maria*, 20/03/81), (*pedro*,05/09/75), (*carlos*, 10/10/88)].
Deve imprimir: pedro, maria, joao, carlos

Solução:

```
imp_por_idade(L):- arruma_lista(L,L1), ordena(L1,[],L2), imp_nome(L2), !.

arruma_lista([],[]).
arruma_lista([(N,D/M/A)|R1],[(A/M/D,N)|R2]):- arruma_lista(R1,R2), !.

ordena([], L, L).
ordena([X|R], L, L1):- insere(X,L,L2), ordena(R,L2,L1), !.

insere(X,[],[X]).
insere(X, [Y|R], [Y|R1]):- X@< Y, insere(X, R,R1).
insere(X, [Y|R], [X,Y|R]).
```

```
imp_nome([]).
imp_nome([(_,N)|R]):- write(N), nl, imp_nome(R).
```

3. [25 **pts**] **Predicados de coleta de soluções:**
Dado o banco de clausulas abaixo:

```
pais(joao,maria,carlos,12/03/85).
pais(joao,maria,rafael,14/05/86).
pais(joao,claudia,clarisse,25/12/90).
pais(joao,claudia,pedro,18/05/92).
pais(joao,marlene,roberto,18/08/95).
pais(joao,marlene,rita,21/03/97).
pais(joao,marlene,georgina, 05/07/98).
```

(a) [10 **pts**] Faça um predicado que cria uma lista de filhos do joao, separados por casamento, usando o predicado "findall". Neste caso: [[*carlos*,*rafael*],[*clarisse*,*pedro*],[*roberto*,*rit*

Solução:

```
filhos(X,F):- findall(Y,pais(X,Y,_,_),L), list_to_set(L,L1), filhos(X,L1,F), !.
filhos(X,[],[]).
filhos(X,[Y|R1],[F|R2]) :- findall(Z,pais(X,Y,Z,_),F), filhos(X,R1,R2), !.
```

(b) [15 **pts**] Faça a mesma coisa usando o predicado "bagof"
Solução: Questão anulada, todos ganham os 15 pontos.

4. [60 **pts**] **Leitura/Gravação:**
Dado um arquivo de texto com o formato abaixo:

```
5
1234 "joao da silva" "engenheiro civil" 10/03/2007 7200.55
1235 "mauro santos" "carpinteiro" 05/05/2008 1024.33
1246 "alfredo kordeiro" "servente" 07/03/2001 743.25
1387 "claudia ribeiro" "escrituraria" 08/11/2005 1525.99
1366 "roberta ajambuja" "presidente" 02/01/1998 25725.99
```

Onde o primeiro valor especifica o número de funcionários de uma empresa, e cada linha contem os dados de um funcionário (código, nome, profissão, data admissão, salário). Faça:

(a) [15 **pts**] Um predicado que lê o arquivo e cria um banco de dados (não de clausulas) com os dados lidos.

Solução:

```
le_arquivo(X):-  open(X,read,F), read_number(F,N),  grava(F,N), !.

grava(_,0).
grava(F,N):- le_cliente(F,C), recordz(cliente,C), N1 is N - 1, grava(F,N1), !.
```

```
le_cliente(F,(C,N,P,A,S)):- read_number(F,C), read_string(F,N),
               read_string(F,P), read_data(F,A), read_number(F,S),!.

read_data(F,D/M/A):- read_number(F,D),read_number(F,M),read_number(F,A), !.

read_number(F,N):- limpa(F), read_number(F,[],N1), number_codes(N,N1), !.

read_number(F,N,N1):- get_code(F,C), (testa_fim(C), N1 = N | poe_fim(C,N,N2),
               read_number(F,N2,N1)), !.

read_string(F,N):- limpa(F), get_code(F,C), testa_sep(C), read_string(F,[],N1),
               atom_codes(N,N1), !.

read_string(F,N,N1):- get_code(F,C), (testa_sep(C), N1 = N | poe_fim(C,N,N2),
read_string(F,N2,N1)), !.

poe_fim(C,[],[C]).
poe_fim(C,[X|R1],[X|R2]):- poe_fim(C,R1,R2), !.

testa_fim(C):- espaco(C).
testa_fim(47).
testa_fim(eof).

testa_sep(34).

limpa(F):- peek_code(F,C), espaco(C), get_code(F,C), limpa(F).
limpa(_).

espaco(32).
espaco(10).
```

(b) Um predicado que implementa uma interface capaz de, pelo menos:

- [10 **pts**]listar os funcionários

  Solução:

```
interface:-  imp_menu, read(O),  chama(O), interface, !.
interface.

imp_menu:- write("Escolha uma das operações abaixo:"), nl,
           write("1 - Lista funcionários"), nl,
           write("2 - Lista dados do funcionario"), nl,
           write("3 - inclui novo funcionário"), nl, !.

chama(1):- recorded(cliente,(C,N,P,A,S),X),
           imp_lista(['Cod: ',C,'  Nome: ',N, '  Profissão: ',P,
                      '  Data Admissão: ',A, '  Salario: ',S]), fail.
chama(1).
```

```
imp_lista([]):- nl, !.
imp_lista([X|R]):- write(X), imp_lista(R), !.
```

- [10 **pts**]listar os dados de um funcionário pelo seu código

  Solução:

```
chama(2):- write('Entre codigo do cliente:'), nl, read(C),
           recorded(cliente,(C,N,P,A,S),X),
           imp_lista(['Cod: ',C,'  Nome: ',N, '  Profissão: ',P,
                      '  Data Admissão: ',A,'  Salario: ',S]),
           nl, nl, !.
```

- [15 **pts**] incluir um novo funcionário.

  Solução:

```
chama(3):- write('Entre com os dados do cliente:'), nl, write("Código:"),
           read(C), write("Nome:"), read(N),  write("Profissão:"), read(P),
           write("Data:"), read(A),  write("Salário:"), read(S),
           recordz(cliente,(C,N,P,A,S),X), nl,
           imp_lista(['Cod: ',C,'  Nome: ',N, '  Profissão: ',P,
                      '  Data Admissão: ',A, '  Salario: ',S]),
           nl, nl, !.
```

(c) [10 **pts**] Um predicado que salva o banco de dados para o arquivo no mesmo formato de leitura.

Solução:

```
salva_arquivo(X):-  open(X,write,F),  findall(C,recorded(cliente,C,_),L),
    length(L,N), write(F,N), nl(F), grava_cliente(F,L), !.

grava_cliente(F,[]).
grava_cliente(F,[(C,N,P,A,S)|R]):-   write(F,C),
     write(F,' '), writeq(F,N), write(F,' '), writeq(F,P), write(F,' '),
     write(F,A), write(F,' '),write(F,S), nl(F), grava_cliente(F,R), !.
```

Boa Prova