Nome: Gustavo Hammerschmidt.

Lista 7 -Grafos

PUCPR

1) (10%) Implemente um grafo não direcionado que representa os caminhos existentes entre as capitais do Brasil. Considere que existe uma estrada ligando duas capitais quando os respectivos estados fazem fronteira. Utilize o mapa na folha seguinte.

```
% Questão 1: Caminhos entre as capitais.
%
% O Predicado cc (Capital) relaciona
% os caminhos entre as capitais e confere um
% peso aos caminhos entre essas capitais.
%
%  Predicado | Capital |  Capital  | Peso
%     cc (  'exemploX','Exemploy',  Z).
%
cc('Porto_Alegre','Florianópolis',1).
cc('Florianópolis','Porto_Alegre',1).
cc('Florianópolis','Curitiba',1).      % Região Sul.
cc('Curitiba','Florianópolis',1).
cc('Curitiba','São_Paulo',1).
cc('Curitiba','Campo_Grande',1).

cc('Campo_Grande','Cuiabá',1).
cc('Campo_Grande','Goiânia',1).
cc('Campo_Grande','Belo_Horizonte',1).
cc('Campo_Grande','São_Paulo',1).
cc('Cuiabá','Porto_Velho',1).
cc('Cuiabá','Manaus',1).
cc('Cuiabá','Belém',1).
cc('Cuiabá','Palmas',1).          % Região Centro-Oeste.
cc('Cuiabá','Goiânia',1).
cc('Cuiabá','Campo_Grande',1).
```

```
cc('Goiânia','Campo_Grande',1).
cc('Goiânia','São_Paulo',1).
cc('Goiânia','Belo_Horizonte',1).
cc('Goiânia','Salvador',1).
cc('Goiânia','Palmas',1).
cc('Goiânia','Cuiabá',1).
cc('Goiânia','Distrito_Federal',1).
cc('Distrito_Federal','Goiânia',1).
cc('Distrito_Federal','Belo_Horizonte',1).

cc('São_Paulo','Campo_Grande',1).
cc('São_Paulo','Belo_Horizonte',1).
cc('São_Paulo','Rio_de_Janeiro',1).
cc('São_Paulo','Curitiba',1).
cc('Rio_de_Janeiro','São_Paulo',1).
cc('Rio_de_Janeiro','Vitória',1).
cc('Rio_de_Janeiro','Belo_Horizonte',1).
cc('Vitória','Salvador',1).          % Região Sudeste.
cc('Vitória','Belo_Horizonte',1).
cc('Vitória','Rio_de_Janeiro',1).
cc('Belo_Horizonte','Vitória',1).
cc('Belo_Horizonte','Rio_de_Janeiro',1).
cc('Belo_Horizonte','São_Paulo',1).
cc('Belo_Horizonte','Goiânia',1).
cc('Belo_Horizonte','Salvador',1).
cc('Belo_Horizonte','Distrito_Federal',1).

cc('Rio_Branco','Porto_Velho',1).
cc('Rio_Branco','Manaus',1).
cc('Manaus','Rio_Branco',1).
cc('Manaus','Boa_Vista',1).
```

```
cc('Manaus','Porto_Velho',1).
cc('Manaus','Belém',1).
cc('Manaus','Cuiabá',1).
cc('Boa_Vista','Manaus',1).
cc('Boa_Vista','Belém',1).
cc('Macapá','Belém',1).
cc('Belém','Macapá',1).
cc('Belém','Boa_Vista',1).
cc('Belém','Manaus',1).          % Região Norte.
cc('Belém','Cuiabá',1).
cc('Belém','Palmas',1).
cc('Belém','São_Luís',1).
cc('Porto_Velho','Manaus',1).
cc('Porto_Velho','Cuiabá',1).
cc('Porto_Velho','Rio_Branco',1).
cc('Palmas','Belém',1).
cc('Palmas','São_Luís',1).
cc('Palmas','Teresina',1).
cc('Palmas','Salvador',1).
cc('Palmas','Goiânia',1).
cc('Palmas','Cuiabá',1).

cc('Salvador','Belo_Horizonte',1).
cc('Salvador','Goiânia',1).
cc('Salvador','Palmas',1).
cc('Salvador','Terezina',1).
cc('Salvador','Aracaju',1).
cc('Salvador','Maceió',1).
cc('Salvador','Recife',1).
cc('Aracaju','Salvador',1).
cc('Aracaju','Maceió',1).
```

```
cc('Maceió','Salvador',1).
cc('Maceió','Recife',1).
cc('Maceió','Aracaju',1).        % Região Nordeste.
cc('Recife','Maceió',1).
cc('Recife','Salvador',1).
cc('Recife','Terezina',1).
cc('Recife','Fortaleza',1).
cc('Recife','João_Pessoa',1).
cc('Terezina','Salvador',1).
cc('Terezina','São_Luíz',1).
cc('Terezina','Fortaleza',1).
cc('Terezina','Recife',1).
cc('Terezina','Palmas',1).
cc('São_Luíz','Palmas',1).
cc('São_Luíz','Terezina',1).
cc('São_Luíz','Belém',1).
cc('Fortaleza','Terezina',1).
cc('Fortaleza','Recife',1).
cc('Fortaleza','Natal',1).
cc('Fortaleza','João_Pessoa',1).
cc('Natal','Fortaleza',1).
cc('Natal','João_Pessoa',1).
cc('João_Pessoa','Fortaleza',1).
cc('João_Pessoa','Natal',1).
cc('João_Pessoa','Recife',1).

% ////////////////////////////////////////////////////////
```

2) (20%) Implemente um predicado que percorre o grafo em PROFUNDIDADE e verifica se uma cidade X pode ser alcançada a partir de uma cidade Y retornando o caminho percorrido (nós visitados) em uma lista.

```
% Função de busca em profundidade.
% caminho(Início,Fim,Percurso,Lista a retornar,Contador,Distância).
caminho(No,No, _,[No],D,D).

caminho(A,Z,Percurso,[A|Fim],Count,D):-
   cc(A,Prox,Peso), % Pega o nó filho de A.
   not(member(Prox,Percurso)), % Verifica se já não está no Percurso.
   caminho(Prox,Z,[A|Percurso],Fim,Count+Peso,D).%Faz o mesmo Processo para o nó filho.


% Retorna o primeiro caminho encontrado.
% A,Z -> Capitais; Lista -> Caminho de A Z.
quest_2(A,Z,Lista):-
   caminho(A,Z,[1],Lista,0,_), % Input -> A, Z,[1], 0.
   !.  % Output -> Lista.
```

3) (30%) Implemente um predicado que percorre o grafo em PROFUNDIDADE e verifica se uma cidade X pode ser alcançada a partir de uma cidade Y retornando o caminho percorrido (nós visitados) em uma lista e a distância percorrida no caminho. Considere que uma ligação entre os nós X e Y corresponde a uma distância 1 entre X e Y.

```
% Retorna o primeiro caminho encontrado e a distância.
% A,Z -> Capitais; Lista -> Percurso; D -> Distância.
quest_3(A,Z,Lista,D):-
   caminho(A,Z,[1],Lista,0,X),!, % Input -> A,Z,[1],0.
   D is X. % Output -> Lista,D.
```

4) (20%) Implemente um predicado que percorre o grafo em LARGURA e verifica se uma cidade X pode ser alcançada a partir de uma cidade Y retornando o caminho percorrido (nós visitados) em uma lista.

```
% Função de busca em largura.
% caminho4(Destino,[n(Destinho, Percurso)]|_],Nós_visitados,
%                                   Caminho_reverso).
caminho4(Meta,[n(Meta,Caminho)|_],_,Rcaminho):-  reverse(Rcaminho,Caminho).

caminho4(Meta,[n(Inicio,CI)|RCI],Visitados,Caminho):-
   write('-------------------------------------'),nl,
   write('Meta: '),write(Meta),tab(4),write('Início: '),write(Inicio),nl,
   write('Caminho desde o Início: '),write(CI),nl,nl,

   write('Nós filhos anteriores: '),write(RCI),nl,nl,

                % Pega o próximo de início se não estiver em visitados
   findall(n(I1,[CI,[Inicio]]),(cc(Inicio,I1,_),\+member(I1,Visitados)),Cs),
                % Depois adiciona a uma lista e a retorna.

   write('Nós Filhos: '),tab(4),write(Cs),nl,nl,
   append(RCI,Cs,Nc), % Novo Começo é Lista do findall + Calda(RCI).
   write('Novo Caminho: '),tab(4),write(Nc),nl,nl,
   write('Nós Visitados: '),tab(4),write([Inicio,Visitados]),nl,nl,
   caminho4(Meta,Nc,[Inicio|Visitados],Caminho). % chama para nós filhos.


% função auxiliar ao predicado quest_4/3.
edit([X|Y],Meta,SS):-
   Z = [Y|X],      % organiza lista.
   append(Z,Meta,Z1), % coloca o destino.
   flatten(Z1,SS).    % achata a lista.
```

```
% Retorna o Caminho entre Início e Meta.
% Início, Meta -> Capitais; Caminho -> Caminho de Início à Meta.
quest_4(Inicio,Meta,Caminho):-
   caminho4(Meta,[n(Inicio,[])],[],Aux), % Input -> Meta, n(Início,[]),[].
   edit(Aux,Meta,Caminho),!. %  Output -> Caminho.
```

5) (20%) Implemente um predicado que retorne as distâncias possíveis entre dois nós do grafo. Note que pode haver mais de uma distância em função dos vários caminhos possíveis entre os nós.

```
% Obs.: Questão 5 utiliza o mesmo predicado das questões 2 e 3:
% caminho/6.

% vetor das distâncias possíveis. Retorna todos os caminhos.
vek(A,Z,Y):-
   findall([Lista,D],(caminho(A,Z,[1],Lista,0,X),D is X),Y).

% função para organizar o output do predicado quest_5/2.
organizar([[X|Y]|Z]):-
   write('Caminho: '),write(X),nl,  % printa o percurso.
   write('Distância: '),write(Y),nl, % printa a distância.
   nl,organizar(Z).

% Retorna todos os caminhos possíveis de A a Z e suas
% respectivas distâncias.
% A,Z -> Capitais.
quest_5(A,Z) :-
   vek(A,Z,Lista), % Input -> A,Z.
   organizar(Lista). % Output -> Lista.
```