% INTELIGÊNCIA ARTIFICIAL Exame, xx/xx/xxxx (Tempo: 3h)

Identificação do aluno:

Nome: _____ #: _____

I

1. Analise as seguintes funções escritas em Python e explique o que fazem:

a)
```
def f(x):
    if x==[]:
        return 0
    if x[0]>0:
        return x[0] + f(x[1:])
    return f(x[1:])
```

```
Recebe uma lista de números como argumento e devolve a soma
de todos os numeros positivos.
```

b)
```
def g(x):
    if x==[]:
        return [[]]
    y = g(x[1:])
    return y + [ [x[0]]+z for z in y ]
```

```
Recebe uma lista e devolve o subconjunto de todos os elementos e
uma lista vazia.
```

2. No contexto da geração de todas as interpretações de uma fórmula em lógica proposicional, é necessário gerar todas as combinações de valores possíveis das diversas variáveis proposicionais contidas na fórmula. Assim, programe uma função que, dada uma lista de variáveis proposicionais, gere todas as combinações de valores possíveis.
Exemplo:
```
>>> interpretacoes(["a","b"])
[ [ ("a",True), ("b",True) ], [ ("a",True), ("b",False) ], [ ("a",False), ("b",True) ], [ ("a",False), ("b",False) ] ]
```

```
def interpretacoes(lst):
    if len(lst) == 0:
        return [[]]
    else:
        return [[(lst[0], True)] + x for x in interpretacoes(lst[1:])]
+ [[(lst[0], False)] + x for x in interpretacoes(lst[1:])]
```

II

1. Neste exercício, tem um conjunto de questões de escolha. Em cada alínea, apenas uma das opções dadas está certa, e apenas pode seleccionar uma delas. Cada resposta errada desconta 20% da cotação da alínea.

a) A frase "Todos os livros de Banda Desenhada têm capa dura" pode ser representada em Lógica de 1ª ordem da seguinte forma:

$\forall x\ (Livro(x) \wedge BandaDesenhada(x)) \Rightarrow \neg\ CapaDura(x)$	
$\forall x\ BandaDesenhada(x) \Rightarrow \neg\ Capa(x,Dura)$	
$\forall x\ Livro(x) \vee (\ BandaDesenhada(x)) \Rightarrow \neg\ Capa(x,Dura)$	
$\forall x\ Livro(x) \wedge (\ BandaDesenhada(x)) \Rightarrow \neg\ Capa(x,Dura)$	
Nenhuma das anteriores	X

b) A frase "A melhor nota a Português foi a da Ana" pode ser representada em Lógica de 1ª ordem da seguinte forma:

$\forall x\ Nota(Ana,Português) > Nota(x,Português) \wedge Aluno(x)$	
$\forall x,y,z\ Nota(Ana,Português,y) > Nota(x,Português,z) \wedge y>z$	
$\forall x\ Nota(Ana,Português) \geq Nota(x,Português) \vee Aluno(x)$	
$\forall x\ Aluno(x) \Rightarrow (\ Nota(Ana,Português) \geq Nota(x,Português)\ )$	
Nenhuma das anteriores	X

c) Pesquisa por melhorias sucessivas é:

Uma técnica de pesquisa para resolução problemas de atribuição	
Uma técnica para combinação de heurísticas	
Uma técnica de pesquisa para optimização de soluções	X
Um caso particular de recozimento simulado em que a evolução da temperatura faz lembrar uma paisagem de montanhas	
Nenhuma das anteriores	

d) Uma consequência lógica do conjunto de fórmulas { $A \vee B, \neg B \vee C \vee D, \neg A, \neg D$} é:

$B \wedge A$	
C	X
A	
$A \vee D$	
Nenhuma das anteriores	

e) Os operadores STRIPS são:

Um formato de representação de acções para sistemas reactivos com estado interno	X
Um formato de representação de transições de estados para pesquisa por melhorias sucessivas	
Mecanismos de modificação da solução em pesquisa por recozimento simulado	
Mecanismos para geração de planos no mundo dos blocos	
Nenhuma das anteriores	

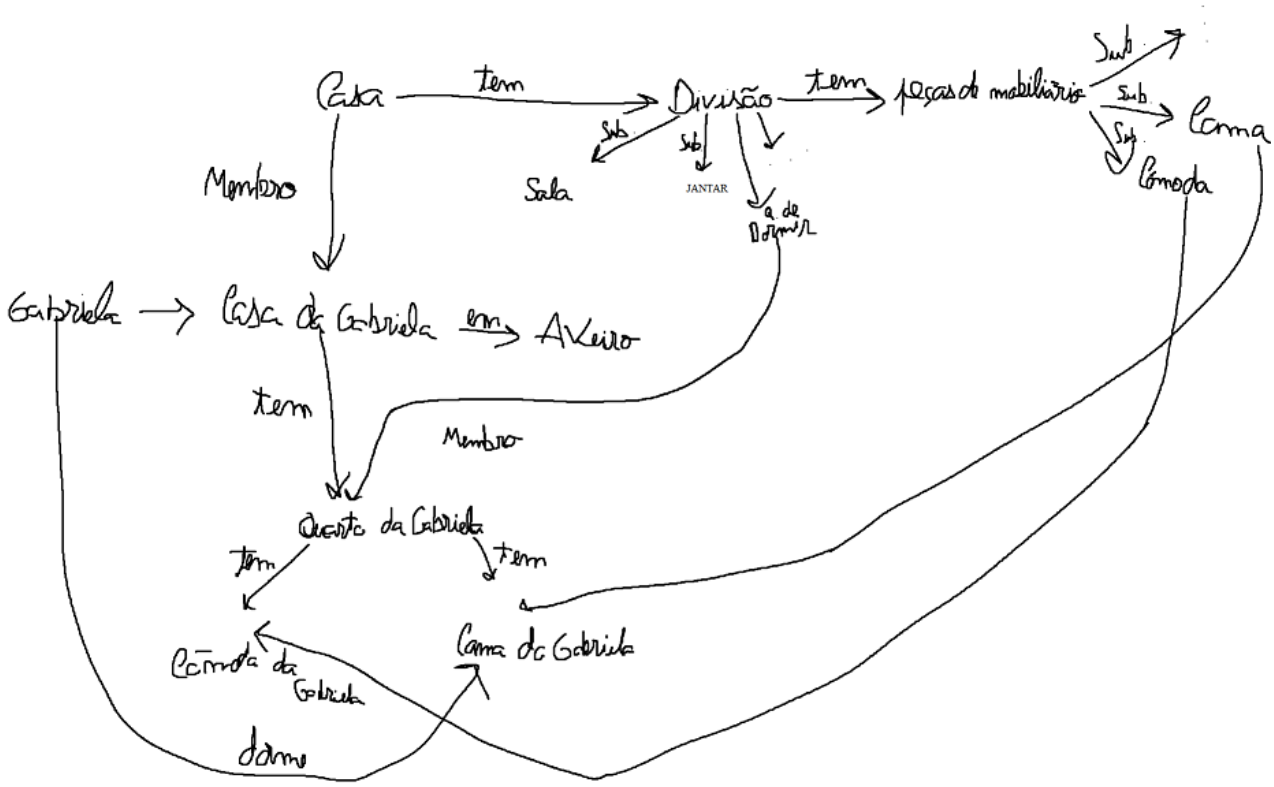
O montanhismo é uma tecnica de pesquisa local que, vai procurar levar--nos de um estado para um outro que seja melhor, em termos de maximizar uma funçao de avaliaçao ou minimizar custos.

2. Identifique semelhanças e diferenças entre a pesquisa em árvore em profundidade e a pesquisa por montanhismo.

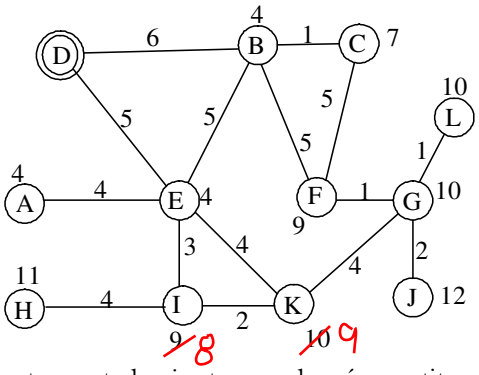
O Depth-first é um algoritmo de pesquisa nao informado classico. Como tal, não faz uso de nenhum tipo de informação que determine se certo estado é melhor que outro, limitando-se a expandir sempre o nó mais profundo ate encontrar a soluçao, chegar a um limite de profundidade maximo ou, o nó mais profundo ja nao possui sucessores (nestes ultimos casos, ele é obrigado a fazer backtracking e expandir um nó noutro nivel menos profundo.
O montanhismo trata entao de problemas de otimizaçao, guardando apenas o estado atual. Depth guarda tudo e procura apenas encontrar uma soluçao.

Sao ambas nao completas (e.g pode ficar preso em maximos locais e o depth pode nao chegar à soluçao(depth infinita ou limite mt baixo).

3. As casas têm divisões de diferentes tipos, por exemplo, salas de estar, salas de jantar, quartos de dormir, cozinhas e quartos de banho. As divisões da casa têm peças de mobiliário, como por exemplo, mesas, cadeiras, camas, cómodas e estantes. A casa da Gabriela é em Aveiro. Essa casa tem um quarto com uma cama, em que a Gabriela dorme, e uma cómoda. Represente este conhecimento através de uma rede semântica.



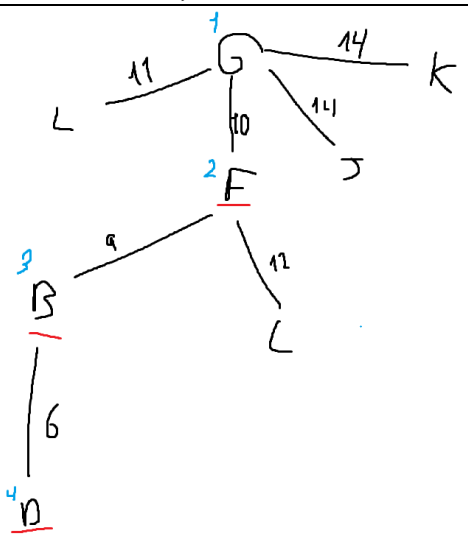
4. O grafo a seguir apresentado representa um espaço de estados num problema de pesquisa, sendo **D** o estado objectivo (solução). As estimativas do custo de chegar à solução a partir de cada estado estão anotadas junto aos mesmos. Os custos das transições estão anotados junto às ligações do grafo.



a) Verifique se as estimativas de custo anotadas junto a cada nó constituem uma heurística admissível para a pesquisa A*. Se não for esse o caso, introduza (na própria figura) alterações que a tornem admissível. Justifique.

```
  A heuristica tem que ser menor ou igual ao custo.
A - Custo: 9 Heuristica: 4   F - Custo: 11 H: 9   K - Custo: 9   H: 10
B - Custo: 6 Heuristica: 4   G - Custo: 12 H: 10  L - Custo: 13 H: 10
C - Custo: 7 Heuristica: 7   I - Custo: 8  H: 9
E - Custo: 5 Heuristica: 4   J - Custo: 14 H: 12
```

b) Tomando o estado **G** como estado inicial, apresente a árvore de pesquisa gerada quando se realiza uma pesquisa A* com repetição de estados. Numere os nós pela ordem em que são acrescentados à árvore e anote também o valor da função de avaliação em cada nó. Em caso de empate nos valores da função de avaliação em dois ou mais nós, utilize a ordem alfabética dos respectivos estados.



5. Considere um veículo autónomo que se movimenta num ambiente estruturado em nós e ligações, ou seja, estruturado como um grafo. As ligações correspondem a ruas. Os nós representam confluências de uma ou mais ruas. Além disso, em cada nó pode haver 0 ou mais parques de estacionamento. As ruas começam e terminam em nós adjacentes no grafo. O agente é capaz de realizar as seguintes acções: atravessar (passar para outra rua do nó), estacionar num dos parques do mesmo nó, percorrer (seguir até ao nó no outro extremo da rua actual), sair do estacionamento para uma dada rua que começa ou termina no mesmo nó.

a) Identifique e caracterize um conjunto de predicados em lógica de primeira ordem que possam ser usados para especificar condições sobre estados de planeamento neste domínio. Identifique os valores possíveis dos argumentos desses predicados. (Nota: Para responder a esta pergunta, é aconselhável ver também a alínea b), onde estes predicados também são usados.)

```
parked(x), x -> Carro estacionado em x.
on(x), x -> Carro não estacionado no nó x.
directed_at(x) -> Carro direcionado para nó x.
has_park(x) -> Nó tem parque(s).
are_connected(x,y) -> x e y sao nós conectados.
```

b) Usando os predicados que propôs, defina um conjunto de operadores STRIPS para representar as acções que podem ser realizadas neste domínio.

```
# x é o nó atual, y
atravessar():
args = [x,y]
pre condiçoes = [on(x), are_connected(x,y)]
neg = [on(x)]
pos = [on(x), directed_at(y)]


# x é o nó atual
estacionar():
args = [x, y]
pre condiçoes = [on(x), has_park(x)]
neg = [on(x)]
pos = [parked(x)]


#
percorrer():
args = [x,y]
pre_condiçoes = [on(x), directed_at(y)]
neg = [on(x), directed_at(y)]
pos = [on(y), directed_at(x)]



sair_estacionamento():
args = [x,y]
pre_condiçoes = [parked(x), are_connected(x,y)]
neg = [parked(x), are_connected(x,y)]
pos = [on(x), directed at(y)]
```

Identificação do aluno:

Nome: _____ #: _____