

Identificação do aluno:

Nome: _____ #: _____

I

1. Analise os predicados escritos em Mercury apresentados em seguida e explique o que fazem:

- a) :- pred primeiro(list(T), list(T), list(T)).
 :- mode primeiro(in, in, out) is det.
 primeiro([],L2,L2).
 primeiro([H|T],L2,[H|T2])
 :- primeiro(T,L2,T2).

- b) :- pred segundo(list(T)::in,list(T)::out) is multi.
 segundo([],[]).
 segundo(L1,L2)
 :- if delete(L1,X,R1) then segundo(R1,R2), L2=[X|R2] else L2=[].

- c) :- pred terceiro(pred(T1,T2),list(T1),list(T2)).
 :- mode terceiro(pred(in,out) is det, in, out) is det.
 terceiro(_,[],[]).
 terceiro(P,[H1|T1],[H2|T2])
 :- P(H1,H2),
 terceiro(P,T1,T2).

2. O módulo cuja interface é apresentada em seguida exporta um tipo abstracto que representa uma tabela dos jogadores com as melhores pontuações e fornece predicados e funções para manipular essas tabelas. Este módulo é para usar no contexto de jogos que, do ponto de vista do utilizador, apresentam uma tabela ordenada por ordem decrescente da pontuação, contendo o nome de um jogador, a sua pontuação e a data em que essa pontuação foi obtida.

Esta tabela tem uma dimensão máxima de NR registos. Para evitar que uma tabela seja dominada pelo mesmo jogador, esta só apresenta no máximo NJ registos por jogador, com $NJ < NR$. Por exemplo, se a tabela só tiver NJ registos do mesmo jogador (não está completa) e este obtiver uma pontuação inferior à menor presente na tabela, aquela não é registada.

Cada entrada da tabela contém:

- O nome do jogador;
- A pontuação;
- A data (dia, mês e ano) em que foi obtida a pontuação.

Exemplo de uma tabela ($NR = 10$ e $NJ = 3$):

João	100	09/01/2007
Pedro	95	11/12/2007
Ana	81	01/12/2007
João	78	24/02/2007
João	76	05/03/2007

Considere o seguinte esqueleto do módulo que define as funcionalidades pretendidas:

```
:- module highscore.

:- interface.

:- import_module list, io.

:- type highscoreTable.

:- func init(
    int, % maximum records
    int % maximum records per player
) = highscoreTable.

:- pred insert(
    highscoreTable::in,
    highscoreTable::out,
    int::in, % score
    string::in, % player name
    int::in, % day
    int::in, % month
    int::in % year
) is det.

:- pred print(
    highscoreTable::in,
    io::di,
    io::uo
) is det.
```

```
:- pred bestScore(
    highscoreTable::in,
    string::in, % player name
    int::out, % score
    int::out, % day
    int::out, % month
    int::out % year
) is semidet.

:- pred print(
    highscoreTable::in,
    io::di,
    io::uo
) is det.

:- func playerScores(
    highscoreTable,
    string % player name
) = list(int).

:- pred merge(
    highscoreTable::in,
    highscoreTable::in,
    highscoreTable::out
) is det.

:- pred purgePlayer(
    highscoreTable::in,
    highscoreTable::out,
    string::in % player name
) is det.
```

a) Defina o tipo **highscoreTable** e outros tipos que julgue necessários para as alíneas seguintes.


b) Implemente a função **init/2** que, dado os parâmetros NR e NJ, devolve uma tabela de pontuações vazia.

c) Implemente o predicado **bestScore/6** que dado o nome de um jogador devolve a melhor pontuação e data em que a obteve.

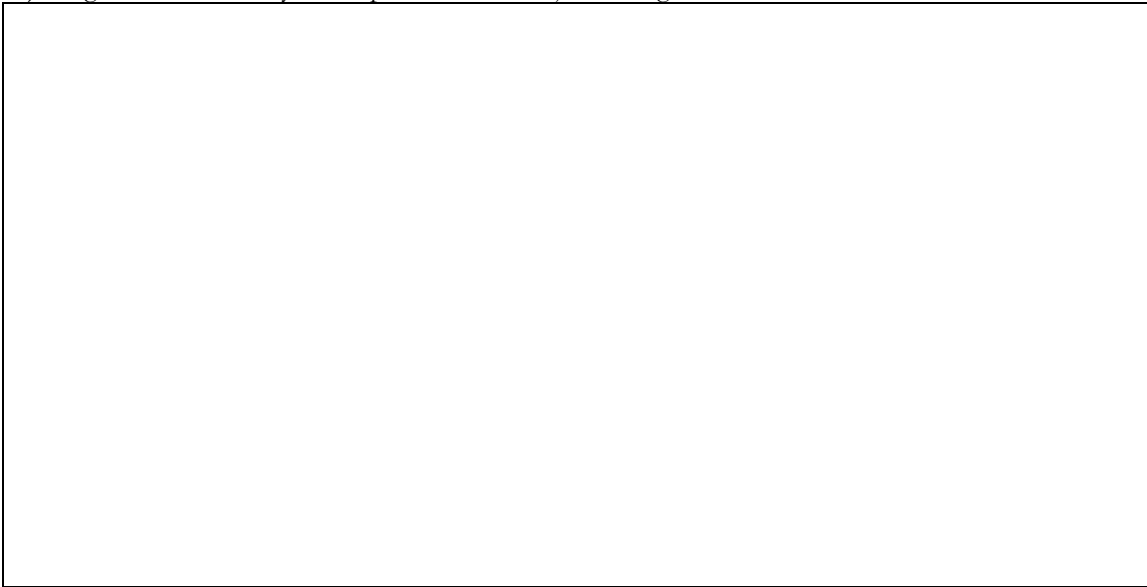
II

1. O robô Nautilus é utilizado na pesca submarina. O Nautilus transporta 10 arpões e um depósito com capacidade para 20 peixes. Quando sente um peixe em frente (condição **peixe_em_frente**) lança imediatamente um arpão (acção **disparar**). Se o arpão atingir um peixe (condição **peixe_atingido**), o robô guarda o peixe no depósito (acção **agarrar**), podendo neste caso recuperar o arpão. Caso contrário, o arpão perde-se. Quando perder todos os arpões, o Nautilus pode reabastecer-se (acção **reabastecer**), ficando novamente com 10 arpões. Quando o depósito de peixes estiver cheio, estes devem ser descarregados (acção **descarregar**), ficando o depósito novamente vazio. Quando não tem mais nada para fazer, limita-se a vaguear (acção **vaguear**). Pretende-se que implemente em Mercury o comportamento deste agente.

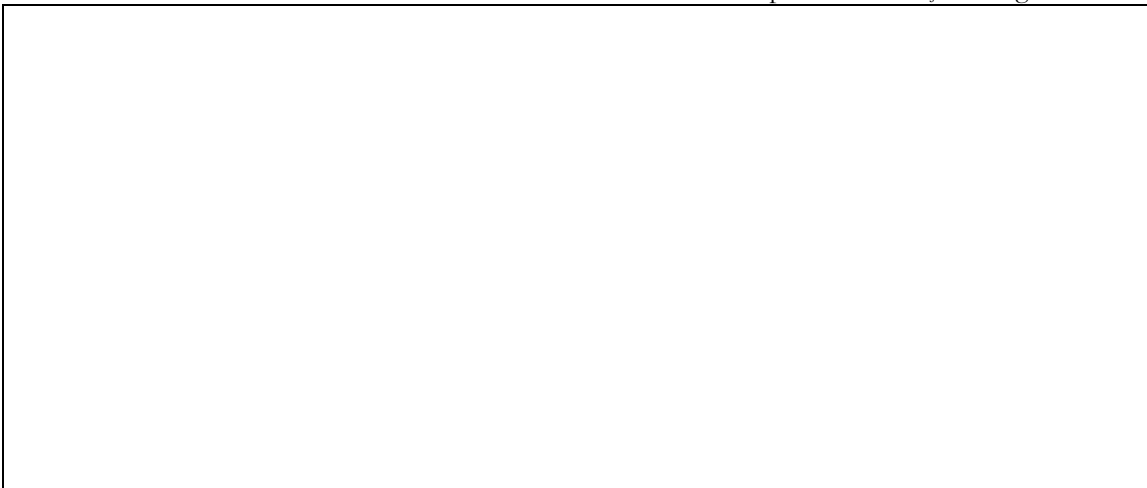
a) Identifique e/ou defina em Mercury os tipos de dados a utilizar.



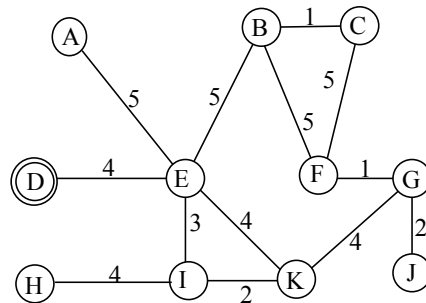
a) Programe em Mercury o comportamento desejado do agente.



2. Enquadre a linguagem KIF no contexto da engenharia do conhecimento, comparando-a com outros formalismos seus conhecidos e comentando a sua relevância para a construção de agentes.



3. Considere que o grafo a seguir apresentado representa um espaço de estados num problema de pesquisa. Os custos das transições estão anotados junto às ligações do grafo.



a) Tomando o estado J como estado inicial e o estado D como solução, apresente as sucessivas árvores de pesquisa geradas, quando se realiza uma pesquisa em profundidade sem repetição de estados e com limite crescente (aprofundamento iterativo). Numere os nós pela ordem em que são acrescentados a cada árvore. O limite inicial da profundidade é 2 [a raiz está na profundidade 0].

b) Calcule o factor de ramificação médio da última árvore gerada, ou seja, da árvore em que se encontrou a solução.

4. O caso particular da pesquisa por recozimento simulado (*simulated annealing*) com temperatura $T = 0$ corresponde a que outra técnica de pesquisa sua conhecida? Justifique.