Trabalho 1: Pesquisa Informada e Não Informada

> *Um robot aspirador tem de limpar uma casa. A bateria do robot é limitada mas pode ser recarregada em carregadores colocados na casa.*
>
> Considere que a casa está representada por uma grelha `N x M`. Cada posição pode estar `suja` ou `limpa`.
>
> Se o robot estiver numa posição `suja` pode fazer a acção `limpar`, que muda o estado dessa posição para `limpa`. As posições `limpas` permanecem limpas. Além disso o robot pode deslocar-se para `cima`, `baixo`, `esquerda` e `direita`.
>
> Cada acção do robot consome uma unidade de energia. Se a carga da bateria for `0` o robot não pode executar qualquer acção. Na casa existem algumas posições `carregador`. Quando o robot ocupa uma dessas posições a sua bateria fica com a `carga_máxima`.
>
> O objectivo é **limpar todas as posições sujas** e **ficar numa posição** `carregador`.

Considere o **estado inicial**

```
! - r - !
- - c - -
! - - - !
```

- A (grelha da) casa tem dimensão `3 x 5` (três linhas e cinco colunas).
- Existe apenas uma posição `carregador`, assinalada por `c`.
- A `carga_máxima` da bateria é `7`.
- O `robot` está na posição assinalada por `r`, com carga máxima.
- As salas `limpas` estão assinaladas por `-`.
- As salas `sujas` estão assinaladas por `!`.

1. Pesquisa Não Informada.

1. **Represente**, em `Prolog`, o espaço de estados e os operadores de transição para este problema.
2. **Apresente** o código, em `Prolog`, do **algoritmo de pesquisa não informada** mais eficiente para resolver este problema.
3. **Justifique** a sua escolha do algoritmo. **Indique**:
    i. O número total (exacto) de estados visitados.
    ii. O número máximo (exacto) de estados que ficaram simultaneamente em memória.

2. Pesquisa Informada.

1. **Proponha** duas heurísticas admissíveis para estimar o custo de um estado até à solução para este problema.
2. **Apresente** o código, em `Prolog`, do algoritmo de pesquisa informada mais eficiente para resolver este problema, usando as heurísticas definidas na alínea anterior. **Indique**, para cada heurística:
    i. O número total (exacto) de estados visitados.
    ii. O número máximo (exacto) de estados que ficaram simultaneamente em memória.

3. [Bónus] Pesquisa Iterativa (*local search*).

1. **Indique** uma representação, em `Prolog`, dos estados deste problema, adequados a uma pesquisa iterativa.
2. **Defina** o operador `vizinho`, que gera todos os estados vizinhos de um estado dado.
3. **Indique** uma função `avaliacao`, que permita ordenar os estados por grau de adequação.
4. **Implemente** o algoritmo `Hill Climbing` e resolva este problema, **indicando** o número de estados expandidos até encontrar a solução.