Projeto

Contexto

Pretende-se construir um programa que permita a um utilizador resolver um puzzle. As regras desse puzzle são as seguintes:

- Cada casa contém um símbolo (uma letra inicialmente minúscula);
- Em cada linha e coluna só pode existir uma única réplica de cada símbolo que é pintada a branco (coloca-se a letra em maiúsculas);
- Todas as outras réplicas desse símbolo têm que ser riscadas (substituídas por um cardinal);
- Se uma casa está riscada, todas as casas vizinhas ortogonais têm que estar pintadas a branco;
- É necessário existir um caminho ortogonal entre todas as casas brancas do tabuleiro.

Eis um exemplo de um tabuleiro inicial:

```
e c a d c
d c d e c
b d d c e
c d e e b
a c c b b
```

E da sua solução:

```
E # A D C
D C # E #
B # D C E
C D E # B
A # C B #
```

Etapas

O projeto é dividido em 5 etapas:

1. Motor básico que permite visualizar o jogo, selecionar uma casa e pintá-la de branco ou riscá-la;

2. Permitir voltar atrás e verificar as restrições das casas brancas e casas riscadas;
3. Verificar que há um caminho ortogonal entre quaisquer duas casas brancas no tabuleiro;
4. Dar dicas e resolver o jogo;
5. Entrega final.

Critérios para cada uma das entregas

- Só são avaliados os alunos presentes aquando da entrega;
- Os alunos devem entregar o código e testes usando o CUnit;
- Os alunos devem ser capazes de explicar cada uma das linhas do código que escreveram;
- Se ninguém for capaz de explicar o código, a avaliação será nula.

O que entregar em cada etapa

- Uma makefile com os targets **jogo** e **testar**
- Ao correr `make testar` devem ser executados todos os testes que devem testar todas as condições do programa;
- Deve ser utilizada uma ferramenta de cobertura de código, e.g. **gcov**, para garantir que todas as linhas de código são cobertas pelos testes;
- Ao correr `make jogo` deve passar a existir um executável chamado **jogo** que permite correr o programa;
- A compilação deve sempre utilizar as flags `-Wall -Wextra -pedantic -O1 -fsanitize=address -fno-omit-frame-pointer -g` e compilar na máquina virtual **sem warnings nem erros**.

Pontuação de cada etapa

Cada etapa vale dois pontos, sendo um dos pontos para a implementação de todas as funcionalidades e o outro para a implementação dos testes que cubram todas as possíveis eventualidades.

Comandos a implementar

O programa deve funcionar segundo o esquema REPL (read, execute, print, loop) e deve implementar os seguintes comandos:

g jogo gravar o estado atual do jogo num ficheiro
l jogo ler o estado do jogo de um ficheiro
<coordenada> no formato <letra minúscula><número> onde a letra corresponde a uma coluna e o número a uma linha

b <coordenada> colocar a letra da casa correspondente à coordenada em maiúsculas

r <coordenada> colocar um # no local da letra

v verificar o estado do jogo e apontar todas as restrições violadas

a ajudar mudando o estado de todas as casas que se conseguem inferir através do estado atual do tabuleiro:

- riscar todas as letras iguais a uma letra branca na mesma linha e/ou coluna
- pintar de branco todas as casas vizinhas de uma casa riscada
- pintar de branco uma casa quando seria impossível que esta fosse riscada por isolar casas brancas

A invocar o comando **a** enquanto o jogo sofrer alterações

R resolver o jogo

d desfazer o último comando executado

s sair do programa