Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: BCC201 – Introdução a Programação
Professores: Túlio A. M. Toffolo e Puca Huachi V. Penna

Trabalho Prático: Resta UM

Quem nunca ouviu falar no jogo *Resta UM*? Eis a descrição na *Wikipédia*:

> Resta um é um quebra-cabeça no qual o objetivo é, por meio de movimentos válidos, deixar apenas uma peça no tabuleiro.
>
> No início do jogo, há 32 peças no tabuleiro, deixando vazia a posição central. Um movimento consiste em pegar uma peça e fazê-la "saltar"sobre outra peça, sempre na horizontal ou na vertical, terminando em um espaço vazio. A peça que foi "saltada"é retirada do tabuleiro. O jogo termina quando não é mais possível fazer nenhum outro movimento. Nesta ocasião, o jogador ganha se restar apenas uma peça no tabuleiro.



Neste trabalho vamos considerar uma versão mais "versátil" do Resta UM em que o tabuleiro pode ter formatos e dimensões variadas, e as peças podem estar posicionadas em diferentes posições. O jogo será visualizado pelo terminal, e o usuário digitará comandos para mover as peças. Capriche para que o jogo seja o mais claro e divertido possível!

O Trabalho Prático

Você deve implementar um programa que lê um arquivo de texto, cujo nome deve ser passado como argumento na execução do programa, contendo os dados do jogo: (i) as dimensões n e m da matriz $n \times m$, e (ii) $n \times m$ números inteiros, indicando a disponibilidade e o conteúdo de cada célula. Note que cada célula pode ter um dos seguintes valores:

- **-1**: indica que a célula não pode ser utilizada;
- **0**: indica que a célula pode ser utilizada mas não tem nenhuma peça;
- **1**: indica que a célula pode ser utilizada e tem uma peça.

O Resta UM padrão considera uma matriz de dimensões 7×7, representado pelo seguinte arquivo:

resta_um.txt

```
7 7

-1 -1  1  1  1 -1 -1
-1 -1  1  1  1 -1 -1
 1  1  1  1  1  1  1
 1  1  1  0  1  1  1
 1  1  1  1  1  1  1
-1 -1  1  1  1 -1 -1
-1 -1  1  1  1 -1 -1
```

Note que os dois números inteiros na primeira linha indicam o tamanho da matriz. Em seguida, temos exatamente $7 \times 7 = 49$ números inteiros no arquivo, identificando a disponibilidade e o conteúdo das 49 células da matriz.

Seu programa deve ser capaz de salvar um jogo em andamento, para que o jogador seja capaz de continuar o jogo mais tarde. Para isso, o programa deve gerar um arquivo indicando a situação atual do jogo. Este arquivo deve seguir exatamente o mesmo formato do arquivo de entrada.

Seguem exemplos de dois arquivos (note que a extensão **.txt** é utilizada). O primeiro (inicio.txt) descreve um jogo no início, ou seja, antes que o jogador faça qualquer jogada. O segundo arquivo (quase.txt), descreve um jogo em andamento, após várias jogadas.

inicio.txt

```
7 7

-1 -1  1  1  1 -1 -1
-1 -1  1  1  1 -1 -1
 1  1  1  1  1  1  1
 1  1  1  0  1  1  1
 1  1  1  1  1  1  1
-1 -1  1  1  1 -1 -1
-1 -1  1  1  1 -1 -1
```

quase.txt

```
7 7

-1 -1  1  1  1 -1 -1
-1 -1  1  0  1 -1 -1
 1  1  1  1  1  1  1
 1  1  1  1  0  0  1
 1  1  1  1  1  1  1
-1 -1  1  0  1 -1 -1
-1 -1  1  0  1 -1 -1
```

Execução do programa

Seu programa, logo após ser executado, deverá imprimir o tabuleiro do *Resta UM*. Não utilize números inteiros. Você escolhe como imprimirá o tabuleiro, portanto **capriche na visualização** (ou nem mesmo os professores da disciplina vão querer jogar o jogo).

Em seguida, o usuário será convidado a **digitar comandos** para fazer movimentos no jogo, salvar o jogo, ou mesmo finalizá-lo (veja possíveis comandos na Tabela 1). Após cada comando, você deve reimprimir o tabuleiro com as eventuais modificações.

Importante: o usuário deve ser alertado com uma mensagem de erro caso tente fazer um movimento que infrinja as regras do jogo. Por exemplo, se o usuário digitar um comando inválido ou tentar mover uma peça em uma direção para a qual ela não pode ser movida, você deve informá-lo qual engano ele cometeu e solicitar que ele digite um comando novamente.

Seu programa deve informar ao usuário se ele **perdeu** ou **venceu**.

Tabela 1: Lista de comandos possíveis

Comando	Argumento	Resultado
c	CD	Move um pino na célula da linha C e coluna D para **cima**.
b	CD	Move um pino na célula da linha C e coluna D para **baixo**.
e	CD	Move um pino na célula da linha C e coluna D para a **esquerda**.
d	CD	Move um pino na célula da linha C e coluna D para a **direita**.
ajuda	n	Seu programa deve sugerir e executar n movimentos para o jogador.
salvar	out	Salva o jogo tal como está no momento no arquivo "out.txt".
sair		Encerra o programa (sem salvar as últimas alterações).

Exemplo de execução do programa (dados digitados pelo usuário estão destacados em **azul**):

```
./resta_um padrao.txt

Bem vindo ao Resta UM!

Abrindo o jogo padrao.txt:

      A B C D E F G
    A     O O O
    B     O O O
    C O O O O O O O
    D O O O . O O O
    E O O O O O O O
    F     O O O
    G     O O O

Digite um comando: b BD

      A B C D E F G
    A     O O O
    B     O . O
    C O O O . O O O
    D O O O O O O O
    E O O O O O O O
    F     O O O
    G     O O O

Digite um comando: b AD
Comando/movimento invalido!

Digite um comando: b GF
Comando/movimento invalido!

Digite um comando: ajuda 1
Executando o movimento 1: "c ED"

      A B C D E F G
    A     O O O
    B     O . O
    C O O O O O O O
    D O O O . O O O
    E O O O . O O O
    F     O O O
    G     O O O

Digite um comando: salvar meu_jogo.txt
Jogo salvo com sucesso no arquivo meu_jogo.txt

```

```
Digite um comando: ajuda 2
Executando o movimento 1: "d DB"

      A B C D E F G
    A     O O O
    B     O . O
    C O O O O O O O
    D O . . O O O O
    E O O O . O O O
    F     O O O
    G     O O O

Executando o movimento 2: "c DD"

      A B C D E F G
    A     O O O
    B     O O O
    C O O O . O O O
    D O . . . O O O
    E O O O . O O O
    F     O O O
    G     O O O

Digite um comando: salvar meu_jogo.txt
Jogo salvo com sucesso no arquivo meu_jogo.txt

Digite um comando: sair
```

Note que a estética do jogo de vocês deve diferir do exemplo acima – que inclusive não é nada bonito, justamente para não limitá-los. O objetivo do exemplo é apenas mostrar algumas interações com o usuário. Tais interações devem funcionar da mesma forma no programa de vocês.

Desafio (1)

- Se o usuário não especificar um arquivo de texto com o jogo de entrada, crie um jogo aleatório.
- O objetivo é gerar jogos interessantes, de diferentes níveis de dificuldade, de forma aleatória e gerar um arquivo texto contendo o jogo gerado!

Desafio (2)

- Seu programa obrigatoriamente deve informar ao usuário se ele **perdeu** ou **venceu**.
- No entanto, quanto antes o programa avisar ao usuário que ele perdeu o jogo, melhor. O desafio é tentar detectar o quanto antes!!
- Capriche no algoritmo de detecção! Afinal, não queremos que o usuário perca seu tempo, certo?

Instruções

- O problema deve ser resolvido por meio de um programa em C.
- Não serão aceitos trabalhos que caracterizem cópia (mesma estrutura e algumas pequenas modificações) de outro.
- Após a entrega dos trabalhos serão marcadas entrevistas com cada um dos alunos para apresentação dos mesmos para um dos professores da disciplina.

Entrega

- A entrega do código-fonte será feita pelo Moodle.
- Também deverá ser entregue um breve relatório sobre o trabalho contendo:
  - Descrição do problema tratado (não copiar este enunciado).
  - Relato das dificuldades encontradas durante a realização do trabalho e soluções encontradas.
  - Referências de sites e outros materiais utilizados para confecção de trabalhos, incluindo consultas a colegas (especificar quais).
- Atenção: o código-fonte do programa não deve ser incluído no relatório.
- **Vocês deverão também entregar um vídeo de 3 a 5 minutos explicando o código e mostrando seu programa.**

Avaliação

- Funcionamento adequado do programa.
- Atendimento ao enunciado do trabalho.
- Clareza do código (que deve ser devidamente comentado e indentado).
- Utilização de funções.
- Adequação da estrutura do programa (variáveis e comandos utilizados).
- Apresentação do trabalho e relatório.
- Compilação (códigos que não compilam serão zerados, e *warnings* diminuirão a nota).