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

Trabalho Prático: **Sumplete**

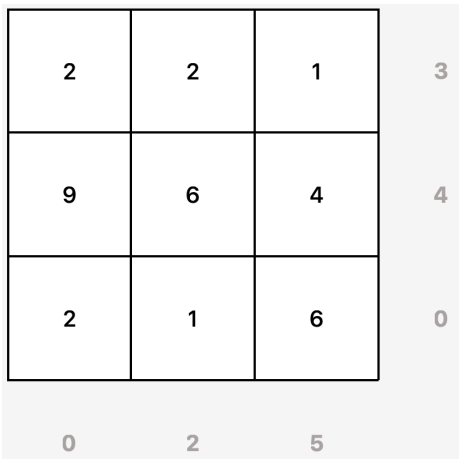
Como jogar Sumplete

Sumplete é um jogo de números baseado em lógica criado recentemente em colaboração com ChatGPT. Ele é inspirado em *Sudoku*, *Kakuro* e *Hitori*, mas com algumas diferenças.

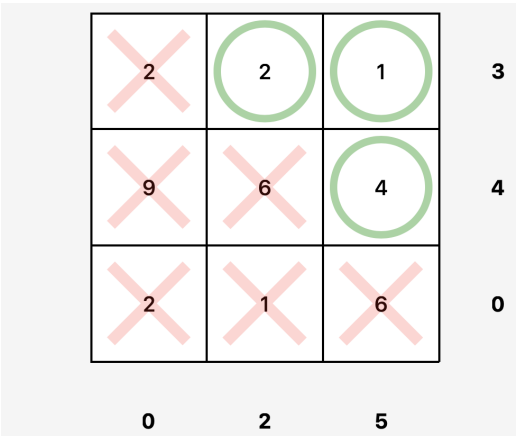
No Sumplete, o jogo consiste de uma tabuleiro, $n \times n$, onde cada célula possui números entre 1 e 9. A diferença é que cada linha e coluna também tem uma "dica" associada a ela. Esta dica informa a soma dos números naquela linha ou coluna.

A Figura 1 mostra um jogo 3×3, os números do lado direito e abaixo do tabuleiro indicam as dicas das linhas e colunas, respectivamente.

Por exemplo, a dica de soma da linha 1 é 3. Portanto, os números nessa linha devem somar 3, como é possível observar o Sumplete resolvido da Figura 1b.



(a) Estado inicial do jogo



(b) Estado final do jogo

Figura 1: Jogo Sumplete

Por padrão, cada número na grade está em um estado "desconhecido". Seu objetivo é identificar quais números excluir e quais manter para garantir que a soma em cada coluna e linha corresponda à "dica" correspondente.

Na versão *on-line* do jogo (`https://sumplete.com`), para marcar uma célula como excluída, clique ou toque uma vez e você verá um X. Para marcar uma célula como uma que você com certeza deseja manter, clique ou toque novamente e você verá um círculo verde. Clique ou toque uma última vez para redefinir para o estado "desconhecido".

Você precisará acompanhar as somas de cada linha e coluna à medida que avança para ter certeza de que está no caminho certo.

Depois de remover com sucesso os números corretos, a soma total à direita/inferior acenderá.

Quando todos os números à direita e na parte inferior da grade estiverem acesos, você terá vencido o jogo!

O Trabalho Prático

Você deve implementar o jogo para um jogador utilizando uma matriz para armazená-lo, e deve ser possível continuar um jogo previamente salvo.

O programa deve possuir as seguintes características:

- A cada jogada, o programa deverá solicitar a posição que o jogador deseja marcar (veja o exemplo baixo) e imprimir o tabuleiro atualizado.
- Deve tratar a entrada de dados, como não deixar um jogador marcar uma posição inválida ou digitar um comando inválido.
- A interface não precisa ser gráfica ou sofisticada, mas o jogador deve ser capaz de entender o estado do jogo rapidamente a cada jogada.
- Avisar quando um jogador ganhar o jogo e mostrar seu tempo de jogo.

Para resolver o problema, escolha a representação que considerar mais conveniente (vetor ou matriz).

Não utilize variáveis globais no programa. Faça funções para marcar a posição selecionada pelo jogador e também para efetuar todas as operações e verificações necessárias. Pode-se utilizar passagem de parâmetros por valor ou referência nas funções.

Seu programa deve ser executado seguindo este fluxo:

1. O programa deve exibir um menu com as opções (0) para sair, (1) começar um novo jogo, (2) continuar um jogo salvo, (3) voltar para o jogo em andamento, ou (4) exibir o ranking.
   (a) Se a opção for iniciar um novo jogo, o programa deve solicitar o tamanho do tabuleiro e o nível de dificuldade, respectivamente
      - as dimensões podem ser: 3, 4, 5, 6, 7, 8 ou 9;
      - o nível de dificuldade pode ser **F**ácil, **M**édio ou **D**ifícil
   (b) Se a opção for continuar um jogo salvo, o programa deve solicitar o nome do arquivo texto contendo o estado inicial e imprimir o jogo logo em seguida. Continue do passo 2.

Figura 2: Estados das células do jogo

(c) Se a opção for continuar o jogo atual, o programa volta para o jogo. Observe que essa opção só pode ser selecionada se o usuário já tiver começado.

(d) Se a opção exibir o ranking for selecionado, o programa deve imprimir uma lista com o ranking dos jogadores. Os dados serão lidos de um arquivo, conforme descrito abaixo.

2. O jogador deve digitar um dos comandos a seguir, até finalizar o jogo ou voltar para o menu inicial.

manter: para marcar uma posição que entra na soma. O usuário deve digitar junto com o comando a linha e coluna que deseja marcar, da seguinte forma `manter <lin><col>`. Ex.: `manter 12`. Marca a linha um, coluna 2.

remover: para remover uma posição da soma. O usuário deve digitar junto com o comando a linha e coluna que deseja marcar, da seguinte forma `remover <lin><col>`. Ex.: `remover 12`. Marca a linha um, coluna 2.

dica: o programa deve marcar uma posição que entra na soma.

resolver: resolve o jogo do ponto que o usuário parou.

salvar: para armazenar em um arquivo o jogo em seu estado atual. O usuário deve digitar, obrigatoriamente, o comando seguido do nome do arquivo. Exemplo: "`salvar jogo.txt`" salva o jogo no arquivo "jogo.txt".

voltar: volta para o menu inicial.

Importante: seu programa deve proibir que o usuário execute comandos inválidos. O usuário deve ser alertado com uma mensagem de erro caso digite um valor inválido. O programa também deve detectar quando um jogo acaba com vitória de algum jogador ou empate.

Arquivo de configuração

O jogo deve permitir armazenar os resultados de até 5 jogadores para cada tamanho de tabuleiro mostrando-os ao ser selecionada a opção correspondente no menu inicial, para isso as seguintes funcionalidades devem ser implementadas ao jogo:

- Manter um arquivo de configuração (denominado `sumplete.ini`) que permita ler e gravar as informações dos jogadores;
- Sempre que uma partida finalizar mostrar o tempo que o jogador gastou e a posição do jogador no ranking;
  - o ranking dever estar ordenado do menor tempo para o maior
- Ao finalizar o programa o arquivo de configuração deve ser atualizado.

Exemplo do arquivo `sumplete.ini` com 2 tamanhos da tabuleiro jogados:

```
size = 3
player1 = John Silva
time1 = 12
player2 = Paul Souza
time2 = 34
player3 = Computador
time3 = 55
player4 = Ringo Oliveira
time4 = 56
player5 = George Lucas
```

```
time5 = 1977

size = 4
player1 = Astro Jetson
time1 = 3
player2 = Rosie Jetson
time2 = 4
```

O arquivo possui 5 jogadores no tamanho três tipos de informações, o tamanho do tabuleiro, o nome do jogador e seu tempo, cada informação é precidida da palavra chave *size*, *playerN* e *timeN*, respectivamente. onde N indica a posição do jogador no ranking.

Entre cada palavra chave e seu valor existe um simbolo de igual. Pode ou não haver espaços entre esse simbolo. Da mesma forma pode ou não ter linha em branco no arquivo.

Arquivo com o jogo

O arquivo de um jogo que será lido/escrito contém na linha 1 a dimensão d do jogo. A seguir, as d linhas matriz são apresentadas, contendo os valores das d colunas separados por espaços. A linha imediatamente após contém a soma das linhas e na linha abaixo a soma das colunas. Em seguida, contém a quantidade m de valores marcados para manter. Nas m linhas seguintes são apresentadas a linha e coluna de cada marcação. Logo após as m linhas é apresentado o número r de valores removidos, seguidas das linhas com as posições. Por fim, o arquivo apresenta o nome do jogador e o seu tempo de jogo, um em cada linha.

O quadro a seguir indica um exemplo de arquivo contendo um jogo.

```
3
2 2 1
9 6 4
2 1 6
3 4 6
0 2 5
1
2 3
2
1 1
3 2
Arthur Dent
5
```

O arquivo acima apresenta o jogo da Figura 1a, com a célula (2, 3) marcada para entrar na soma e as células (1,1) e (3, 2) removidas da soma. O nome do jogador é Arthur Dent e já jogou por 5 segundos.

Instruções

- O trabalho é individual.
- O problema deve ser resolvido por meio de um programa em C.

- Inclua seu nome e número de matrícula como comentário em todos os arquivos .c e .h gerados.
- 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 os professores.

Entrega

- Você deverá entregar o código fonte (arquivo ZIP).
- A entrega deve ser feita pelo Moodle.
- As entrevistas serão feitas nos horários da aulas.

Avaliação

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

Exemplo de Execução

A seguir é exibido um exemplo, como referência, não é necessário segui-lo. Faça sua interface como achar mais interessante.

Você pode (e deve) customizar e melhorar as saídas do programa. A seguir segue um exemplo simples apenas para entendimento (os dados digitados pelo usuário estão destacados em azul):

Exemplo:

./sumplete

Bem vindo ao Jogo SUMPLETE

0. Sair do Jogo
1. Começar um novo jogo
2. Continuar um jogo salvo em arquivo
3. Continuar o jogo atual
4. Exibir o ranking

Durante o jogo digite "voltar" para retornar ao menu.

Escolha a opção: **1**

Digite o nome do jogador 1: **Boris**

Digite o tamanho do tabuleiro (3 à 9): **3**

Digite o nível de dificuldade (Fácil, Médio ou Difícil): **F**

	1	2	3	
1	5	4	5	5
2	6	7	1	0
3	4	1	5	9
	9	0	5	

Boris, digite o comando: **manter 11**

	1	2	3	
1	5	4	5	5
2	6	7	1	0
3	4	1	5	9
	9	0	5	

Boris, digite o comando: **remover 12**

	1	2	3	
1	5	4	5	5
2	6	7	1	0
3	4	1	5	9
	9	0	5	

Boris, digite o comando: **remover 13**

	1	2	3	
1	5	4	5	5
2	6	7	1	0
3	4	1	5	9
	9	0	5	

Continua ...