Prova 2

Algoritmos e Estruturas de Dados I

Professor: Pedro O.S. Vaz de Melo

1cm
Nome: _____

escrevendo o meu nome eu juro que seguirei o código de honra

Código de Honra para este exame:

- Não darei ajuda a outros colegas durante os exames, nem lhes pedirei ajuda;
- não copiarei nem deixarei que um colega copie de mim;
- não usarei no exame elementos de consulta não autorizados.

Informações importantes:

- Em questões que pede um **programa**, este deve ser completo, com bibliotecas (incluindo, quando necessário, a biblioteca `prova.h`), função main, etc. Se deve ser feita uma **função**, somente a função é suficiente. Se deve ser feito um **procedimento**, somente o procedimento é suficiente.
- A interpretação das questões da prova faz parte do critério de avaliação. Caso tenha dúvida sobre a sua interpretação de uma determinada questão, escreva as suas suposições na resolução da mesma.
- As funções implementadas no módulo `prova1.h` podem ser usadas em **qualquer** exercício da prova. Além disso, se você usar uma função do módulo `prova.h`, considere que ela está implementada de forma correta.

1. (*5 points*) Implemente uma função que recebe uma matriz de inteiros $n \times n$ e o seu tamanho n e a preenche com 0s em todas as suas posições. O protótipo dessa função deve ser:

```
int zeraMatriz(int M[ ][10], int n);
```

2. (*5 points*) Implemente uma função que recebe uma matriz de inteiros $n \times n$ zerada e o seu tamanho n e atribui 1 aleatoriamente a n das suas células. Importante: garanta que exatamente n células tenham seus valores alterados para 1. O protótipo dessa função deve ser:

```
int initMatriz(int M[ ][10], int n);
```

3. (*12 points*) As questões abaixo referem-se a um programa que simula uma partida do jogo *Batalha Naval*. Nesse jogo, inicialmente cada jogador deve posicionar 10 navios em um tabuleiro (matriz) 10×10. Depois, os jogadores intercalam ataques, sendo que um ataque é um palpite sobre a posição $(linha, coluna)$ que um navio pode estar no tabuleiro do adversário. Se o palpite for correto, então o navio do adversário é destruído. Vence o jogo aquele que conseguir destruir todos os navios do adversário.

a. (*2 pts*) Defina um **novo tipo de dados** para representar um `Jogador` e que deve ser capaz de armazenar as seguintes informações: `id` (código do jogador), `n` (número de palpites feitos), `navios` (número de navios restantes no tabuleiro) e `M`. `M` é o tabuleiro do jogador, sendo uma matriz de inteiros que informa em qual linhas e colunas o jogador possui navios. Considere que se `M[i][j]==1`, então o jogador tem um navio na linha i e coluna j. Se `M[i][j]==0`, então não há navios nessa célula.

b. (*4 pts*) Implemente uma função de nome `initJogador` que recebe um `Jogador` **por referência** e o seu identificador (inteiro) e inicializa os seus campos `id`, `n`, `navios` e `M`. Considere que inicialmente os jogadores possuem 10 navios e que M deve ser preenchida aleatoriamente. Dica: use as funções dos exercícios 1 e 2 para preencher M.

c. (*5 pts*) Escreva uma função de nome `ataca` que recebe dois `Jogadores` **por referência**, $j1$ e $j2$, e simula um ataque de $j1$ em $j2$. Neste ataque, $j1$ escolhe uma posição aleatória da matriz M do jogador $j2$ para atacar. Caso tenha um navio nesta posição, destrua o navio. Lembre-se de atualizar os campos `n` do $j1$ e `navios` de $j2$ (caso necessário).

d. (*3 pts*) Complete o código abaixo.

```
#include <stdio.h>

#include _________________________________
void main() {
   Jogador j1, j2;

   //inicializa jogadores 1 e 2:

   initJogador(_____________,1);

   initJogador(_____________,2);

   //enquanto o jogador 1 estiver vivo:

   while(_________________________________) {

      //jogador 1 ataca 2:

      ataca(_________________________________);

      //se o jogador 2 estiver vivo:

      if(_________________________________

         //jogador 2 ataca 1:

         ataca(_________________________________);

      else {

         printf("\nVencedor: %d (%d ataques)", j1.id, j1.ntrials);

         return;
      }
   }
   printf("\nVencedor: %d (%d ataques)", j2.id, j2.ntrials);
}
```