Departamento de Informática – PUC-Rio
INF1620 - Estruturas de Dados

Segunda Lista de Exercícios – 2005.1

1. Implemente uma função que receba uma string e um número inteiro `n` como parâmetros, e retorne uma nova string com os `n` primeiros caracteres da string passada como parâmetro. Por exemplo, recebendo como parâmetros a string "Estruturas" e o número 3, essa função retornaria uma nova string contendo a sequência de caracteres "Est". Essa função deve obedecer o protótipo:

   ```
   char* prefixo (char* str, int n);
   ```

   Obs.: A string passada como parâmetro não pode ser alterada. Assuma que `n` é sempre menor do que o comprimento da string passada como parâmetro.

2. Implemente uma função que receba uma string como parâmetro e retorne uma nova string com os caracteres minúsculos trocados para maiúsculos e vice-versa. Caracteres que não forem letras devem ser copiados sem alteração para a nova string. Por exemplo, se for passado como parâmetro a string "PUC-Rio", essa função deve retornar uma nova string contendo a sequência de caracteres "puc-rIO". Essa função deve obedecer o protótipo:

   ```
   char* inverte_letra (char* str);
   ```

   Obs.: A string passada como parâmetro não pode ser alterada.

3. Implemente uma função que receba como parâmetro uma string e dois caracteres (`original` e `novo`), e retorne uma nova string com todas as ocorrências do caractere `original` substituídas pelo caractere `novo`. Por exemplo, se recebendo como para parâmetro a string "Estruturas" e os caracteres 't' e 'd', essa função deve retornar uma nova string contendo a sequência de caracteres "Esdruduras". Essa função deve obedecer o protótipo:

   ```
   char* troca_letra (char* str, char original, char novo);
   ```

   Obs.: A string passada como parâmetro não pode ser alterada.

4. Implemente uma função que receba uma string como parâmetro e retorne uma nova string com as letras da string original substituídas por suas sucessoras no alfabeto. Por exemplo, recebendo como parâmetro a string "Casa", essa função retornaria a string "Dbtb". Essa função deve obedecer o protótipo:

   ```
   char* shift_string (char* str);
   ```

   Obs.: A letra 'z' deve ser substituída pela letra 'a' (e 'Z' por 'A'). Caracteres que não forem letras devem ser copiados para a nova string sem sofrer alteração. A string passada como parâmetro não pode ser alterada.

5. Implemente uma função que receba uma string como parâmetro e retorne uma nova string com todas as ocorrências de uma letra substituídas pelo seu *oposto* no alfabeto, isto é, 'a' $\mapsto$ 'z', 'b' $\mapsto$ 'y', 'c' $\mapsto$ 'x', etc. Caracteres que não forem letras devem ser copiados sem alteração para a nova string. Essa função deve obedecer o protótipo:

```
char* string_oposta (char* str);
```

   Obs.: A string passada como parâmetro não pode ser alterada.

6. Implemente uma função que receba uma string como parâmetro e retorne uma nova string que seja a string do parâmetro de trás para frente. Por exemplo, recebendo como parâmetro a string "Aluno", essa função retornaria a string "onulA". Essa função deve obedecer o protótipo:

```
char* inverte_string (char* str);
```

   Obs.: A string passada como parâmetro não pode ser alterada.

7. Considerando a estrutura

```
struct Ponto {
   int x;
   int y;
};
```

   para representar um ponto em uma grade 2D, implemente uma função que indique se um ponto p está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo v_1 e superior direito v_2. A função deve retornar 1 caso o ponto esteja localizado dentro do retângulo e 0 caso contrário. Essa função deve obedecer o protótipo:

```
int dentroRet (struct Ponto* v1, struct Ponto* v2, struct Ponto* p);
```

8. Considerando a estrutura

```
struct Ponto {
   int x;
   int y;
};
```

   para representar um ponto em uma grade 2D, implemente uma função que indique se um ponto p está localizado dentro ou fora de um círculo. O círculo é definido por seu centro c e seu raio r. A função deve retornar 1 caso o ponto esteja localizado dentro do círculo e 0 caso contrário. Essa função deve obedecer o protótipo:

```
int dentroCirculo (struct Ponto* c, int raio, struct Ponto* p);
```

9. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que calcule a soma de dois vetores. Essa função deve obedecer o protótipo:

```
void soma (struct Vetor* v1, struct Vetor* v2, struct Vetor* res);
```

onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem somados, e o parâmetro res é um ponteiro para uma estrutura vetor onde o resultado da operação deve ser armazenado.

10. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que calcule a subtração de dois vetores. Essa função deve obedecer o protótipo:

```
void sub (struct Vetor* v1, struct Vetor* v2, struct Vetor* res);
```

onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem subtraídos, e o parâmetro res é um ponteiro para uma estrutura vetor onde o resultado da operação $v1 - v2$ deve ser armazenado.

11. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que calcule o produto escalar de dois vetores. Essa função deve obedecer o protótipo:

```
float dot (struct Vetor* v1, struct Vetor* v2);
```

onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem multiplicados, e o resultado da operação $v1 \cdot v2$ deve ser retornado.

12. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que multiplique os componentes de um vetor por um número escalar. Essa função deve obedecer o protótipo:

```
void mult (struct Vetor* v, float escalar);
```

onde o parâmetros v é um ponteiro para o vetor a ser multiplicado e *escalar* é o valor pelo qual os componentes do vetor devem ser multiplicados. Observe que o vetor apontado pelo ponteiro v tem seu conteúdo alterado por essa função.

13. Considerando as declarações

```
struct aluno {
  char nome[81];
  char matricula[8];
  char turma;
  float p1;
  float p2;
  float p3;
};
typedef struct aluno Aluno;
```

para representar o cadastro de alunos de uma disciplina, implemente uma função `imprime_reprovados` que imprima o número de matrícula, o nome, a turma e a média de todos os alunos que **não** foram aprovados na disciplina.

Inclua as definições de estrutura **exatamente como acima** no seu arquivo de resposta, mas não inclua a função `main`. Inclua também a referência a quaisquer arquivos de biblioteca que você use (e.g., `stdio.h`).

Os dados de cada aluno reprovado na disciplina devem ser impressos em uma única linha, seguindo o formato

```
numero_de_matricula nome_do_aluno turma media
```

As linhas descritas acima devem ser a **única coisa impressa pela sua função**. Não imprima cabeçalhos, terminadores, separadores ou qualquer outro tipo de decoração que fuja ao padrão especificado. Caso nenhum aluno se encaixe na busca, não imprima nada.

Assuma que o critério para aprovação é

$$\frac{p_1 + p_2 + p_3}{3} >= 5.0$$

Essa função deve obedecer o protótipo

```
void imprime_reprovados (Aluno** turmas, int n);
```

onde o parâmetro `turmas` representa um vetor de ponteiros para estruturas do tipo `Aluno` e o parâmetro `n` é o comprimento do mesmo. Obs.: podem haver posições não utilizadas no vetor `turmas`, contendo o valor `NULL`.

14. Considerando as declarações

```
struct aluno {
  char nome[81];
  char matricula[8];
  char turma;
  float p1;
  float p2;
  float p3;
};
typedef struct aluno Aluno;
```

para representar o cadastro de alunos de uma disciplina, implemente uma função `imprime_media` que imprima o número de matrícula, o nome, a turma e a média de todos os alunos que tiveram a média de suas três provas maior ou igual a um determinado valor (passado como parâmetro).

Inclua as definições de estrutura **exatamente como acima** no seu arquivo de resposta, mas não inclua a função `main`. Inclua também a referência a quaisquer arquivos de biblioteca que você use (e.g., `stdio.h`).

Os dados de cada aluno com nota acima da média especificada devem ser impressos em uma única linha, seguindo o formato

```
numero_de_matricula nome_do_aluno turma media
```

As linhas descritas acima devem ser a **única coisa impressa pela sua função**. Não imprima cabeçalhos, terminadores, separadores ou qualquer outro tipo de decoração que fuja ao padrão especificado. Caso nenhum aluno se encaixe na busca, não imprima nada.

A média de cada aluno é calculada pela fórmula

$$\frac{p_1 + p_2 + p_3}{3}$$

Essa função deve obedecer o protótipo

```
void imprime_media (Aluno** turmas, int n, float media);
```

onde o parâmetro `turmas` representa um vetor de ponteiros para estruturas do tipo `Aluno`, o parâmetro `n` é o comprimento do mesmo, e o parâmetro `media` é o valor da média mínima que queremos usar para selecionar os dados dos alunos que devem ser impressos. Obs.: podem haver posições não utilizadas no vetor `turmas`, contendo o valor `NULL`.

15. Considerando as declarações

```
struct aluno {
  char nome[81];
  char matricula[8];
  char turma;
  float p1;
```

```
  float p2;
  float p3;
};
typedef struct aluno Aluno;
```

para representar o cadastro de alunos de uma disciplina, implemente uma função `imprime_turma` que imprima o número de matrícula, o nome, a turma e a média de todos os alunos que pertencem a uma determinada turma (passada como parâmetro).

Inclua as definições de estrutura **exatamente como acima** no seu arquivo de resposta, mas não inclua a função `main`. Inclua também a referência a quaisquer arquivos de biblioteca que você use (e.g., `stdio.h`).

Os dados de cada aluno da turma devem ser impressos em uma única linha, seguindo o formato

```
numero_de_matricula nome_do_aluno turma media
```

As linhas descritas acima devem ser a **única coisa impressa pela sua função**. Não imprima cabeçalhos, terminadores, separadores ou qualquer outro tipo de decoração que fuja ao padrão especificado. Caso nenhum aluno se encaixe na busca, não imprima nada.

A média de cada aluno é calculada pela fórmula

$$\frac{p_1 + p_2 + p_3}{3}$$

Essa função deve obedecer o protótipo

```
void imprime_turma (Aluno** turmas, int n, char turma);
```

onde o parâmetro `turmas` representa um vetor de ponteiros para estruturas do tipo `Aluno`, o parâmetro `n` é o comprimento do mesmo, e o parâmetro `turma` é a turma que queremos usar para selecionar os dados dos alunos que devem ser impressos. Obs.: podem haver posições não utilizadas no vetor `turmas`, contendo o valor `NULL`.

16. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz identidade ou não. A função deve retornar 1 se a matriz for uma matriz identidade, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores simples**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

    ```
    int matriz_identidade (int* mat, int n);
    ```

17. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz identidade ou não. A função deve retornar 1 se a matriz for uma matriz identidade, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores de ponteiros**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

    ```
    int matriz_identidade (int** mat, int n);
    ```

18. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz triangular superior ou não. A função deve retornar 1 se a matriz for uma matriz triangular superior, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores simples**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int triangular_superior (int* mat, int n);
```

19. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz triangular superior ou não. A função deve retornar 1 se a matriz for uma matriz triangular superior, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores de ponteiros**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int triangular_superior (int** mat, int n);
```

20. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz triangular inferior ou não. A função deve retornar 1 se a matriz for uma matriz triangular inferior, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores simples**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int triangular_inferior (int* mat, int n);
```

21. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz triangular inferior ou não. A função deve retornar 1 se a matriz for uma matriz triangular inferior, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores de ponteiros**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int triangular_inferior (int** mat, int n);
```

22. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz tridiagonal ou não, isto é, somente os elementos da diagonal principal e das duas diagonais adjacentes podem ter valores diferentes de zero:

$$se\ (i \neq j)\ e\ (i \neq j+1)\ e\ (i \neq j-1) \Rightarrow a_{i,j} = 0$$

A função deve retornar 1 se a matriz for uma matriz tridiagonal, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores simples**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int tridiagonal (int* mat, int n);
```

23. Implemente uma função que indique se uma matriz quadrada de números inteiros é uma matriz tridiagonal ou não, isto é, somente os elementos da diagonal principal e das duas diagonais adjacentes podem ter valores diferentes de zero:

$$se\ (i \neq j)\ e\ (i \neq j+1)\ e\ (i \neq j-1) \Rightarrow a_{i,j} = 0$$

A função deve retornar 1 se a matriz for uma matriz tridiagonal, e 0 caso contrário. A função recebe como parâmetros a matriz de inteiros, usando a representação de matrizes através de **vetores de ponteiros**, e um inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
int tridiagonal (int** mat, int n);
```

24. Implemente uma função que retorne o produto dos elementos da diagonal de uma matriz quadrada de números de ponto flutuante (`float`). A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores simples**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float produto_diagonal (float* mat, int n);
```

25. Implemente uma função que retorne o produto dos elementos da diagonal de uma matriz quadrada de números de ponto flutuante (`float`). A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores de ponteiros**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float produto_diagonal (float** mat, int n);
```

26. Implemente uma função que retorne a soma dos elementos acima da diagonal de uma matriz quadrada de números de ponto flutuante (`float`), isto é, a soma de todos os elementos $a_{i,j}$ onde $i < j$. A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores simples**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float soma_superior (float* mat, int n);
```

27. Implemente uma função que retorne a soma dos elementos acima da diagonal de uma matriz quadrada de números de ponto flutuante (`float`), isto é, a soma de todos os elementos $a_{i,j}$ onde $i < j$. A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores de ponteiros**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float soma_superior (float** mat, int n);
```

28. Implemente uma função que retorne a soma dos elementos abaixo da diagonal de uma matriz quadrada de números de ponto flutuante (`float`), isto é, a soma de todos os elementos $a_{i,j}$ onde $i > j$. A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores simples**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float soma_inferior (float* mat, int n);
```

29. Implemente uma função que retorne a soma dos elementos abaixo da diagonal de uma matriz quadrada de números de ponto flutuante (`float`), isto é, a soma de todos os elementos $a_{i,j}$ onde $i > j$. A função recebe como parâmetros a matriz de números de ponto flutuante, usando a representação de matrizes através de **vetores de ponteiros**, e um número inteiro `n`, indicando a dimensão da matriz. Essa função deve obedecer o protótipo:

```
float soma_inferior (float** mat, int n);
```