INF 1037 - Programação em C - 01/10/2025

Questões de Revisão para a P1 - Turmas 33A, 33B, 33C, 33D e 33E

Observação:

- Para algumas questões, faz parte do entendimento do enunciado a definição do protótipo adequado para suas funções.
- Não esqueça de desenvolver os testes para suas funções.

Questão 01) Escreva uma única função que recebe um vetor de inteiros e calcula duas somas: a soma dos valores pares e a soma dos valores ímpares. Além do vetor e da quantidade inteiros no vetor, a função deve receber ponteiros de variáveis onde as somas serão armazenadas.

Questão 02) Escreva uma única função que aceita como parâmetros um vetor de inteiros *v* com *n* valores, onde *v* e *n* são parâmetros da função, e determina o menor elemento do vetor e o número de vezes que este elemento ocorreu no vetor. Por exemplo, para um vetor com os seguintes elementos: 5, 2, 15, 3, 7, 2, 15, 8, 6, 15, a função terá como resultado o número 2 e o valor 2 (indicando que 2 é o menor número do vetor e que o mesmo ocorreu 2 vezes). O valor mínimo e a quantidade de vezes devem ser informados em ponteiros também recebidos como parâmetros na função.

Questão 03) Escreva uma única função que calcula a média aritmética e a média ponderada de um vetor de float de tamanho qualquer. Além do vetor com os valores, a função recebe um segundo vetor com os pesos da média ponderada. Ambas as médias devem ser passadas como parâmetros da função a serem preenchidos.

Questão 04) Escreva uma função que recebe um vetor de inteiros e retorna um novo vetor, alocado dinamicamente, eliminando os elementos repetidos em sequência. Por exemplo, dado {1,2,2,1,4,5,6,6,5} , a função deve criar o vetor {1,2,1,4,5,6,5}. A função deve retornar o novo vetor e receber um parâmetro para preencher a quantidade de elementos inseridos no novo vetor.

Questão 05) Escreva uma função para criar uma nova cadeia de caracteres sem repetição em sequência dos caracteres de uma cadeia fornecida. Por exemplo, dado "AAnnnaa", a função cria e retorna a cadeia "Ana".

Questão 06) Escreva uma função que recebe duas strings e retorna se a segunda é prefixo da primeira. Retorne 1 se for prefixo e 0 caso contrário. Considere que uma string vazia é prefixo da outra. Considere também que duas strings iguais são prefixo uma da outra.

Questão 07) Escreva uma função que dado um nome, retorna uma nova cadeia com o último sobrenome. Por exemplo, dado "Carlos Drumond de Andrade", a nova cadeia deve ser "Andrade".

Questão 08) Escreva uma função que recebe um vetor de strings com nomes de pessoas e retorne um novo vetor de strings, alocado dinamicamente, onde cada elemento, também alocado dinamicamente, é o primeiro nome de cada pessoa. Por exemplo, se for passado para a função o vetor {"Alberto Einsten", "Leonardo da Vinci", "Alan Turing"}, sua função deve retornar o vetor {"Alberto", "Leonardo", "Alan"}.

Dica: Escreva uma função auxiliar que recebe uma string representando o nome completo de uma pessoa e retorne uma nova string, alocada dinamicamente, com o primeiro nome da pessoa.

Questão 09) Escreva uma função para liberar a memória alocada dinamicamente na questão anterior.

Questão 10) Escreva uma função que recebe um vetor de inteiros *v* e a quantidade de elementos *n* nesse vetor e cria um vetor de ponteiros para *char*, contendo strings com a seguinte regra de formação. Se *v[i]* for par, a string deverá conter *v[i]* vezes a letra 'A'. Por outro lado, se *v[i]* for ímpar, a string deverá conter *v[i]*

vezes a letra ‘a’. Como exemplo, para um vetor de entrada *v* = { 2, 4, 3, 1 }, é esperado que seja criado dinamicamente um vetor de char* contendo as strings “AA”, “AAAA”, “aaa” e “a”.

```
char ** converte (int * v , int n);
```

Questão 11) Implemente uma função que recebe como parâmetros dois vetores de inteiros, v1 e v2, e as suas respectivas quantidades de elementos, n1 e n2. A função também deve receber um terceiro vetor v3, com capacidade para (n1 + n2) elementos. Sua função deverá inserir os elementos de v1 e v2 em v3 de modo intercalado. Por exemplo, se v1 = {1, 3, 5, 7} e v2 = {2, 4, 6, 8, 10, 12}, v3 irá conter {1, 2, 3, 4, 5, 6, 7, 8, 10, 12}. Observe que dependendo do número de elementos, os últimos valores podem não ficar intercalados.

```
void intercala(int* v1, int n1, int* v2, int n2, int* v3);
```

Questão 12) Modifique a função da “questão 11” para criar o vetor de retorno dinamicamente. A assinatura da função deve passar a ser:

```
int* intercala2(int* v1, int n1, int* v2, int n2);
```

Questão 13) Escreva uma função que verifica se uma string é um palíndromo e retorna 1 se for ou 0, caso não seja. Uma palavra palíndroma é aquela cuja sequência de letras é simétrica, permitindo uma leitura idêntica da esquerda para a direita ou da direita para a esquerda: ovo, osso, reler, anilina

```
int palindromo(char* texto);
```

Questão 14) Escreva uma função que recebe um vetor de palavras (strings) e uma palavra (string) e retorna o índice do vetor em que a palavra esteja. Se a palavra não for encontrada, a função deve retornar -1.

Questão 15) Escreva uma função que inverte uma string sem criar uma cópia da string original. Se sua função receber a string “pontificia”, a string original deve ser alterada para passar a ser “aicifitnop”.

Questão 16) Considere *strings* que contém o nome completo e a data de nascimento de uma pessoa, separadas por dois pontos. Exemplos dessas *strings* são mostradas abaixo:

“Leonardo da Vinci: 15/04/1452”

“Albert Einstein: 14/03/1879”

Escreva uma função que recebe uma *string* nesse formato e retorne uma nova *string*, alocada dinamicamente, que representa um *login* de usuário. O *login* deve ser formado pelas iniciais do nome (em letras minúsculas), seguido de dois pontos, seguido do ano de nascimento. Por exemplo, para as *strings* acima, sua função deve retornar “ldv:1452” e “ae:1879”, respectivamente.

Questão 17) Escreva uma função que recebe uma *string* com palavras separadas por espaços em branco e retorne um novo vetor de *strings*, alocado dinamicamente, onde cada elemento é uma palavra da *string* original, também alocada dinamicamente. Por exemplo, se a função receber a *string* “Este foi um teste”, a função deve retornar um vetor com 4 *strings*:

Sua função deve retornar o endereço do vetor de *strings* e preencher o endereço recebido com o número de elementos do vetor, seguindo o protótipo:

```
char** split (char* original, int* pNumElem);
```

Questão 18) Escreva a função `char* concatenarInicioNovo(char* s1, char* s2)` que concatena a *string* s2 no **início** da *string* s1 e em uma nova área de memória alocada dinamicamente do tamanho exato. A função retorna a nova *string*.

Por exemplo, para s1 = “galera!” e s2 = “Isto eh tudo, “, a função retorna “Isto eh tudo, galera!”.

O código desta função deve atender às seguintes especificações:

- não usar funções da biblioteca de strings, exceto a `strlen`;
- não usar a notação de índice (ou seja, use obrigatoriamente aritmética de ponteiros);
- a função retorna NULL se ocorrer problema de alocação de memória;
- se ocorrer problema de memória, exibe mensagem e interrompa o processamento na *main*; e
- libere a memória após os testes.

Desenvolva a `main` para testar sua função e, como teste, use os exemplos apresentados. Atenção para as interrupções por problemas de memória e para as ocasiões de liberar memória.

Questão 19) Considere um arquivo texto cuja primeira informação é o número total de registros, um por linha. Cada registro tem duas informações: uma data (que pode ser dia/mês/ano ou mês/ano ou apenas ano, com no máximo 10 caracteres) e um nome de pessoa (com no máximo 80 caracteres).

Por exemplo:

```
4
25/01/1927 Antonio Carlos Jobim
1946 Mario Prata
03/1887  Heitor Villa-Lobos
14/03/38  Glauber Pedro de Andrade Rocha
```

Escreva a função `char** vetorNomes(const char* nomeArq, int* nreg)`, onde `nomeArq` é o nome do arquivo e `nreg` é um endereço de variável que deve armazenar o número de registros. A função deve retornar um vetor de ponteiros para *strings* que são os nomes extraídos da leitura do arquivo texto. Este vetor deve ser alocado dinamicamente do tamanho exato (tanto para o vetor como para cada *string*).

O código desta função deve atender às seguintes especificações:

- No caso de memória insuficiente, a função deve retornar NULL.
- No caso de problemas de abertura de arquivo, a mensagem de erro e a interrupção devem ser tratadas localmente e tão logo ocorram. O caso de problemas de alocação de memória deve ser tratado na `main`.
- No caso de sucesso, com o exemplo de arquivo acima, o número de registros é 4 e o vetor tem Antonio Carlos Jobim na posição 0, Mario Prata na posição 1, … .
- Não se esqueça de fechar o arquivo tão logo não precise mais dele. E, na `main`, **imprima** o vetor e depois **libere** toda a memória (do vetor e das *strings*).

Desenvolva a `main` para testar sua função e, como teste, use os exemplos apresentados. Atenção para as interrupções por problemas de memória e para as ocasiões de liberar memória.

Questão 20) Escreva um programa que:

- Crie/abra um arquivo texto de nome “arq.txt”.
- Permita que o usuário grave diversos caracteres nesse arquivo, até que o usuário entre com o caractere ‘0’.
- Feche o arquivo.

Questão 22) Considere um arquivo texto com as notas dos alunos de uma disciplina. Cada linha do arquivo contém a matrícula de um aluno (cadeia de nove caracteres), seguida pelos valores de suas três notas (P1, P2 e P3). Considere ainda que podem existir linhas em branco no arquivo. Um exemplo desse formato é:

9010087 -2	2.0	4.3	6.5
8820324 -3	7.0	8.2	8.6
9210478 -5	6.0	7.5	7.8
9020256 -8	3.0	0.5	4.2

Escreva uma função que receba como parâmetros o número de matrícula de um aluno e o nome de um arquivo com as notas de uma disciplina no formato descrito, e retorne a média do aluno na disciplina. A média de um aluno é calculada pela fórmula (P1 + P2 + P3)/3. O protótipo da função deve ser:

```
float media (char * mat, char * nome_arquivo );
```

Caso o número de matrícula passado como parâmetro não seja encontrado no arquivo, a função deve retornar –1.0. Se não for possível abrir o arquivo de entrada, a função deve imprimir a mensagem “Erro” e terminar a execução do programa.

Questão 23) Considere a existência de um arquivo texto, denominado “turma.txt”, com um cadastro de uma turma. Para cada aluno da turma, existem duas linhas no arquivo: na primeira linha, encontra-se o nome do aluno; na linha seguinte, encontram-se as duas notas obtidas pelo aluno. Considere que podem existir linhas em branco no arquivo. Um exemplo deste arquivo é mostrado a seguir:

```
Fulano Pereira
9.0 8.0
Beltrano Silva
4.0 5.0

Sicrano Santos
3.0 7.0
Fulana Souza
4.0 4.0

Maria Paula
6.0 7.0
```

Escreva um programa completo que leia o conteúdo do arquivo (“turma.txt”) com o formato anterior e crie outro arquivo, com o nome “aprovados.txt”, com a lista dos alunos que obtiveram médias maiores ou iguais a 5.0, seguindo a mesma ordem do arquivo de entrada. Cada linha do arquivo de saída deve conter o nome e a média obtida pelo aluno. Se esse arquivo anterior fosse usado num exemplo, a saída obtida seria:

Fulano Pereira	8.5
Sicrano Santos	5.0
Maria Paula	6.5

Se o arquivo de entrada não puder ser aberto, deve-se imprimir a mensagem “Erro” na tela e abortar o programa. Pode-se considerar que sempre será possível abrir o arquivo de saída.

Questão 24) Crie um arquivo texto com títulos de livros, seguido de barra (/), seguido do nome do autor ou autores separados por ponto e vírgula (;), como mostra o exemplo abaixo:

```
Dom Casmurro/Machado de Assis
O Nome da Rosa/Umberto Eco
Angustia/Graciliano Ramos
A Menina Que Roubava Livros/Karkus Zusak
O Corpo Fala/Pierre Weil;Roland Tompakow
Viagem ao Mundo dos Taleban/Louriva Santana
Quem Mexeu no Meu Queijo?/Spencer Johnson
```

Nesta tarefa, você vai ler e escrever arquivos textos.

Escreva um programa para ler seu arquivo e gerar um outro arquivo texto com título e autor(es) agrupado por letra inicial do título. A ordem dos títulos em cada grupo deve ser a mesma em que aparecem no arquivo de entrada, mas os grupos devem ser exibidos em ordem alfabética. Caso não existam livros para uma determinada letra, nada será exibido. Cada título exibido deve ter o seguinte formato:

titulo. Autor: *nome do autor* ou

titulo. Autores: *lista de nome dos autores*, separados por ponto e vírgula (;)

Para o arquivo de entrada mostrado acima, o arquivo de saída gerado deve ter o seguinte formato e conteúdo:

```
Titulos iniciados com A:
       Angustia. Autor: Graciliano Ramos
       A Menina Que Roubava Livros. Autor: Karkus Zusak
Titulos iniciados com D:
       Dom Casmurro. Autor: Machado de Assis
Titulos iniciados com O:
       O Nome da Rosa. Autor: Umberto Eco
       O Corpo Fala. Autores: Pierre Weil;Roland Tompakow
Titulos iniciados com Q:
       Quem Mexeu no Meu Queijo?. Autor: Spencer Johnson
Titulos iniciados com V:
       Viagem ao Mundo dos Taleban. Autor: Louriva Santana
```

Na sua implementação, crie, obrigatoriamente, um vetor de cadeias de caracteres para cada título de livro e outro para cada autor(es). Em ambos os vetores use o espaço de memória estritamente necessário para seu armazenamento. Crie funções auxiliares para organizar sua solução. Não se esqueça de liberar a memória alocada dinamicamente quando não for mais necessária.

Assuma que o número máximo de livros no arquivo é 100 e que nenhum título e nem lista de autores excede 127 caracteres. Não use acentuação. Considere apenas títulos começando com letras. Assuma também que não há erro de formato no arquivo de entrada. Se ocorrer erro na abertura de arquivos ou na alocação de memória, o programa deve exibir uma mensagem informativa e ser abortado.

Dica 1: para leitura dos nomes com espaços, use o formato %[...].

Dica 2: para determinar se a leitura foi bem sucedida pode-se usar o valor de retorno da função fscanf.

Questão 25) Considere um arquivo texto de entrada com nomes de pessoas, seguidos de dois pontos (:) e com a profissão logo após. Esse arquivo tem o número de registros disponibilizado na primeira linha. Por exemplo:

```
3
```

```
jose lins do rego: escritor
adriana de oliveira melo: medica
santos dumont: engenheiro
```

Leia este arquivo e construa um vetor de ponteiros para *strings* do tipo "*profissão nome*". Por exemplo, a primeira *string* é "escritor jose lins do rego". Todas as alocações devem ser dinâmicas.

Escreva uma função auxiliar que recebe o ponteiro para o arquivo e retorna a *string* "*profissão nome*" alocada dinamicamente. Retorne NULL quando não mais conseguir ler nomes e profissões (o que vai acontecer no fim do arquivo). Use o fato de que essa função retorna NULL para controlar o *loop* de leitura de linhas na `main`.

Use o vetor de *strings* em uma função que retorna o índice da *string* de maior comprimento. No caso acima, retorna o índice 1. Na *main*, imprima essa *string*. (no exemplo acima, imprime:

```
medica adriana de oliveira melo
```

Grave um arquivo com a strings "profissão nome" na ordem inversa, isto é, da última para a primeira. Faça este exercício usando MÓDULOS (util.h, util.c e meuProg.c). Depois faça um outro programa que lê este arquivo, e exibe as strings na ordem deste arquivo.