Lista de Exercício de Algoritmos –Alocação dinâmica

1) Crie uma estrutura representando um aluno de uma disciplina. Essa estrutura deve conter o número de matrícula do aluno, seu nome e as notas de três provas. Escreva um programa que mostre o tamanho em byte dessa estrutura.

2) Crie uma estrutura chamada Cadastro. Essa estrutura deve conter o nome, a idade e o endereço de uma pessoa. Agora, escreva uma função que receba um inteiro positivo N e retorne o ponteiro para um vetor de tamanho N, alocado dinamicamente, dessa estrutura. Solicite também que o usuário digite os dados desse vetor dentro da função.

3) Elabore um programa que leia do usuário o tamanho de um vetor a ser lido. Em seguida, faça a alocação dinâmica desse vetor. Por fim, leia o vetor do usuário e o imprima.

4) Escreva um programa que aloque dinamicamente uma matriz de inteiros. As dimensões da matriz deverão ser lidas do usuário. Em seguida, escreva uma função que receba um valor e retorne 1, caso o valor esteja na matriz, ou retorne 0, no caso contrário.

5) Escreva uma função que receba um valor inteiro positivo N por parâmetro e retorne o ponteiro para um vetor de tamanho N alocado dinamicamente. Se N for negativo ou igual a zero, um ponteiro nulo deverá ser retornado.

6) Crie uma função que receba uma string e retorne o ponteiro para essa string invertida.

7) Escreva uma função que receba um valor inteiro positivo N por parâmetro e retorne o ponteiro para um vetor de tamanho N alocado dinamicamente. Esse vetor deverá ter os seus elementos preenchidos com certo valor, também passado por parâmetro. Se N for negativo ou igual a zero, um ponteiro nulo deverá ser retornado.

8) Escreva uma função que receba como parâmetro uma matriz A contendo N linhas e N colunas. A função deve retornar o ponteiro para um vetor B de tamanho N alocado dinamicamente, em que cada posição de B é a soma dos números daquela coluna da matriz.

9) Escreva uma função que receba como parâmetro duas matrizes, A e B, e seus tamanhos. A função deve retornar o ponteiro para uma matriz C, em que C é o produto da multiplicação da matriz A pela matriz B. Se a multiplicação das matrizes não for possível, retorne um ponteiro nulo.

10) Considere um cadastro de produtos de um estoque, com as seguintes informações para cada produto:

- Código de identificação do produto: representado por um valor inteiro
- Nome do produto: com até 50 caracteres
- Quantidade disponível no estoque: representado por um nú mero inteiro
- Preço de venda: representado por um valor real

a) Defina uma estrutura em C, denominada produto, que tenha os campos apropriados para guardar as informações de um produto
b) Crie um conjunto de n produtos (n é um valor fornecido pelo usuário) e peça ao usuário para entrar com as informações de cada produto
c) Encontre o produto com o maior preço de venda
d) Encontre o produto com a maior quantidade disponível no estoque

11) Escreva um programa que lê primeiro os 6 números gerados pela loteria na noite de sábado na TV e depois lê seus próprios 6 números. Então, o programa compara quantos números o jogador acertou. Em seguida, ele aloca espaço para um vetor de tamanho igual a quantidade de números

corretos e guarda os números corretos nesse vetor. Finalmente, o programa exibe os números sorteados e os seus números corretos.

12) Faça um programa que simule 'virtualmente' a memória de um computador: o usuário começa especificando o tamanho da memória (define quantos bytes tem a memória), e depois ele irá ter 2 opções: inserir um dado em um determinado endereço, ou consultar o dado contido em um determinado endereço. A memória deve iniciar com todos os dados zerados.

13) Faça um programa que leia dois números N e M e:

- Aloque espaço e leia uma matriz de inteiros N x M;
- Aloque espaço e construa uma matriz transposta M x N de inteiros.
- Mostre as duas matrizes.
- Localize os três maiores números na primeira matriz e mostre a linha e a coluna onde estão.

14) Faça um programa que leia números do teclado e os armazene em um vetor alocado dinamicamente. O usuário irá digitar uma sequência de números, sem limite de quantidade. Os números serão digitados um a um e, sendo que caso ele deseje encerrar a entrada de dados, ele irá digitar o número ZERO. Os dados devem ser armazenados na memória deste modo:

- Inicie com um vetor de tamanho 10 alocado dinamicamente;
- Caso o vetor alocado esteja cheio, aloque um novo vetor do tamanho do vetor anterior adicionado espaço para mais 10 valores (tamanho N+10, onde N inicia com 10);

15) Faça um programa para associar nomes as linhas de uma matriz de caracteres. O usuário ira informar o número máximo de nomes que poderão ser armazenados. Cada nome poderá ter até 30 caracteres com o '\0'. O usuário poderá usar 5 opções diferentes para manipular a matriz:

- Gravar um nome em uma linha da matriz;
- Apagar o nome contido em determinada linha da matriz;
- Informar um nome, procurar a linha onde ele se encontra e substituir por outro nome;
- Informar um nome, procurar a linha onde ele se encontra e apagar;

16) Faça um programa que:

- Peça para o usuário entrar com o nome e a posição (coordenadas X e Y) de N cidades e as armazene em um vetor de estruturas (N e informado pelo usuário);
- Crie uma matriz de distancias entre cidades de tamanho N x N;
- Calcule a distância entre cada duas cidades e armazene na matriz;
- Exiba na tela a matriz de distancias obtida;

Vetor de estruturas e matriz de distâncias devem ser alocados dinamicamente.

17) Crie um programa que implemente o jogo "Bingo de Prog II". Nesse jogo, o jogador deve selecionar a quantidade de números que ele gostaria de apostar (entre 1 e 20), e em seguida, informar os números escolhidos (valores entre 0 e 100). Após receber a aposta, o computador sorteia 20 números (entre 0 e 100) e compara os números sorteados com os números apostados, informando ao apostador a quantidade de acertos e os números que ele acertou.

O seu programa deverá implementar as funções `ler_aposta`, `sorteia_valores` e `compara_aposta`.

A função ler_aposta deve receber como parâmetro a quantidade de números que serão apostados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números apostados. A função deve pedir para o usuário digitar os números apostados e armazena-los no vetor, garantindo que somente números dentro do intervalo de 0 a 100 sejam digitados. A função deve seguir o seguinte protótipo:

```
void ler_aposta(int *aposta, int n);
```

A função sorteia_valores deve receber como parâmetro a quantidade de números que serão sorteados e um vetor previamente alocado dinamicamente para armazenar a quantidade exata de números sorteados. A função deve sortear aleatoriamente os números (entre 0 e 100) e armazená-los no vetor. A função deve seguir o seguinte protótipo:

```
void sorteia_valores(int *sorteio, int n);
```

A função compara_aposta deve receber como parâmetro o vetor com os números apostados (aposta), o vetor com os números sorteados (sorteio), juntamente com os seus respectivos tamanhos (na e ns) e um ponteiro para uma variável inteira (qtdacertos), onde deve ser armazenada a quantidade de acertos. A função deve retornar o ponteiro para um vetor alocado dinamicamente contendo os números que o apostador acertou. A função deve seguir o seguinte protótipo:

```
int* compara_aposta(int *aposta, int *sorteio, int *qtdacertos, int na, int ns);
```

Em seguida, crie a função principal do programa utilizado as funções criadas anteriormente para implementar o jogo "Bingo de Prog II". Lembre-se que os vetores aposta, sorteio e acertos devem ser alocados dinamicamente e a memória alocada deve liberada quando ela não for mais ser utilizada.
Para sortear números aleatórios utilize a função rand da biblioteca stdlib.h. A função rand retorna um número aleatório em um determinado intervalo.

Exemplo:
`x = rand() % 10;` /* x vai receber um valor entre 0 e 10 */

Para garantir que novos números aleatórios sejam sorteados em cada execução do programa é necessário executar a função srand antes de sortear os números. Exemplo:

```
srand(time(NULL));
```

Para poder utilizar essas funções é necessário incluir no programa as bibliotecas stdlib.h e time.h.

Alguns Lembretes:

- Utilize funções;
- Faça a conferência se a memória foi alocada corretamente;
- Libere a memória alocada dinamicamente.