Exercício 01

Escreva uma função para criar vetores de inteiros com diferentes tamanhos. A função deve ser capaz de criar o vetor e preenche-lo com um valor específico.

O nome da função deve ser `criarVetor`.

Entrada:

- tamanho do vetor
- valor utilizado para o preenchimento do vetor

Saída:

- Endereço (heap) da memória onde o vetor foi criado.

Exercício 02

Escreva uma função para duplicar vetores de inteiro. A função deve ser capaz de replicar o conteúdo do vetor em outra região de memória.

O nome da função deve ser `clonarVetor`.

Entrada:

- endereço de memória do vetor original
- tamanho do vetor original

Saída:

- Endereço (heap) da memória da cópia do vetor original.

Exercício 03

Escreva uma função para criar um vetor de inteiros preenchido com valores aleatórios. A função deve ser capaz de criar vetores de diferentes tamanhos.

O nome da função deve ser `criarVetorAleatorio`.

Entrada:

- tamanho do vetor
- valor utilizado para o preenchimento do vetor

Saída:

- Endereço (heap) da memória onde o vetor foi criado.

Exercício 04

Escreva uma função incrementar 1 ao valor de cada elemento do vetor. A função deve percorrer todos os elementos e incrementar o valor.

O nome da função deve ser `incrementaVetor`.

Entrada:

- endereço de memória do vetor
- tamanho do vetor

Saída:

- não tem saída (void)

Exercício 05

Escreva uma função que seja capaz de dobrar o tamanho de um vetor. A função deve preservar os valores do vetor e preencher com zero as posições adicionais e devolver o endereço

Passos para o desenvolvimento da função:

1. Alocar um novo vetor com o dobro do tamanho do vetor recebido por parâmetro.
2. Copiar os valores para o vetor novo
3. Preencher com '0' as novas posições
4. Desalocar o vetor antigo
5. Atualizar a referência do vetor recebido por parâmetro

O nome da função deve ser `dobrar`. A função somente funciona para vetores alocados dinamicamente.

Entrada:

- endereço de memória do vetor
- tamanho do vetor

Saída:

- não tem saída (void)

Exercício 06

Considere o registro definido abaixo

```
typedef struct produto{
  unsigned int codigo;
  char nome[50];
  float preco;
}Produto;
```

a) Crie uma função responsável por criar dinamicamente um produto e inicializar seus valores. Os valores devem ser recebidos como parâmetro.

b) Crie uma função responsável por imprimir um produto. A função deve imprimir os valores de todos os atributos do produto.

c) Escreva exemplos de utilização das funções criadas nos itens 'a' e 'b'

Exercício 07

Escreva uma função para criar uma matriz dinamicamente e preenche-la com um valor específico. A função deve ser capaz de criar uma matriz de qualquer tamanho e preenche-la com qualquer valor.

O nome da função deve ser `criarMatriz`.

Entrada:

- quantidade de linhas
- quantidade de colunas
- valor a ser utilizado no preenchimento da matriz

Saída:

- Endereço (heap) da memória da matriz criada.