Lista de exercícios 09 - Alocação dinâmica

1. Escreva um programa que aloque memória para dois vetores, com tamanhos diferentes, dados pelo usuário. Os vetores devem ser preenchidos com valores aleatórios no intervalo `[0, 10]`. O programa deve então criar um terceiro vetor, capaz de conter todos os elementos `Vi*Vj`, onde `Vi` é um elemento do primeiro vetor e `Vj` é um elemento do segundo vetor. A multiplicação elemento-a-elemento dos dois vetores deve ser calculada, e os valores obtidos devem ser apresentados. Dica: você pode usar uma ou mais funções para organizar o seu código.

2. Em programas que manipulam strings, a leitura de cada string é feita em um buffer com o tamanho máximo para as strings lidas. Por exemplo, o código abaixo lê uma string de até 1024 caracteres (contando o `'\0'` final):

```
#define BUFLEN 1024
(...)
char buffer [BUFLEN];
fgets (buffer, BUFLEN, stdin);
```

   Neste código, o buffer que contém a string possui tamanho 1024, mesmo que a string digitada seja simplesmente "`oi`". A maior parte do espaço não é usada. Além disso, se for necessário ler outra string usando o mesmo buffer, o conteúdo da primeira string digitada será substituído. Nestas situações, uma forma de economizar memória é usar o buffer maior apenas para ler as strings, mas alocar cada nova string em outro espaço, que tem apenas o tamanho necessário.

   Com base nisso, escreva uma função `char* empacotaString (char* string)`, que recebe como parâmetro um buffer contendo uma string e retorna uma cópia da string, mas em um espaço que tem apenas o tamanho necessário. A nova string deve ser alocada dentro da função, mas a responsabilidade de desalocá-la é do chamador.

3. Escreva uma função que recebe como parâmetro um vetor de int. A função deve criar outro vetor, e colocar nele os elementos do vetor passado como parâmetro, mas sem repetições. Por exemplo, se os elementos do vetor original são `[0, 1, 2, 3, 4, 3, 2, 4, 5, 3, 2, 6, 1, 0]`, o novo vetor deve conter apenas `[0, 1, 2, 3, 4, 5, 6]`. O novo vetor deve ter apenas o tamanho necessário para manter os valores. Os dados do vetor original devem ser mantidos intactos. No final, a função deve retornar o número de itens do vetor criado - o vetor em si é retornado como um parâmetro passado por referência.

4. Adapte o exercício da Lista de introdução a Matrizes de forma a solicitar ao usuário número N de linhas do triângulo de Pascal.

5. (a) Construa uma função, denominada `custo_cidades` que, dado um vetor de inteiros com o código das cidades na ordem que deverão ser visitadas, o número de cidades visitadas no percurso e a matriz de custos, retorne o custo total do itinerário. A função deve ter o seguinte protótipo:

   ```
   int custo_cidades (int* cidades, int n_cidades, int** m);
   ```

   (b) a função `main`, que deve

   - solicitar ao usuário a dimensão da matriz de custo,
   - alocar memória para esta matriz utilizando uma função denominada `alocaMatriz`,
   - preencher a matriz com dados digitados pelo usuário,
   - solicitar ao usuário o tamanho do itinerário,
   - alocar memória para o vetor que irá armazenar o itinerário usando uma função denominada `alocaVetor`,
   - preencher o vetor de itinerários
   - chamar a função `custo_cidades` e imprimir na tela o custo.

   Dica: Vale a pena observar que a matriz `m` sempre será quadrada ($n \times n$); e o custo de transporte entre as cidades i e j é dado pelo elemento m_{ij} da matriz. Exemplo: Considerando a seguinte matriz de custos

$$m_{4\times4} = \begin{pmatrix} 17 & 31 & 25 & 1 \\ 23 & 12 & 1 & 8 \\ 75 & 1 & 10 & 400 \\ 1 & 15 & 20 & 33 \end{pmatrix}$$

   O custo do itinerário $\{0, 3, 1, 2, 1, 0\}$ é calculado da seguinte forma: $m_{03}+m_{31}+m_{12}+m_{21}+m_{10} = 1+15+1+1+23 = 41$

6. Vimos que o vazamento de memória é um problema grave, e que mesmo programadores experientes podem deixar de perceber a sua ocorrência. Para testar as consequências deste problema, crie um programa com um loop infinito. Dentro do loop, aloque um vetor de 1024 bytes. Para evitar que o programa rode rápido demais e que otimizações do compilador descartem o vetor, use a função printf para imprimir o conteúdo da primeira posição do vetor (o conteúdo será "lixo" - neste exemplo, o conteúdo do vetor é irrelevante). Não desaloque este vetor. Isso fará com que cada iteração

do loop crie um novo vetor. Acompanhe a utilização de memória usando as ferramentas do sistema operacional. No Windows, pressione `CTRL+ALT+DEL`, escolha a opção de iniciar o Gerenciador de Tarefas, e observe na aba Desempenho o Histórico de Uso da Memória Física; ou na aba Processos o valor associado ao seu programa na coluna Memória. Descreva o que ocorre.

IMPORTANTE: salve todos os seus trabalhos e não deixe nenhum conteúdo importante aberto em algum programa!!!

7. Refaça alguns (ou todos) os exercícios das listas de vetores usando alocação dinâmica.

8. Refaça alguns (ou todos) os exercícios das listas de matrizes usando alocação dinâmica.