EXERCÍCIOS – PONTEIROS, ALOCAÇÃO DINÂMICA DE MEMÓRIA

1. Escreva uma função que receba um vetor de números reais e tenha como valor de retorno um novo vetor, alocado dinamicamente, com os elementos do vetor original em ordem reversa. A função deve ter como valor de retorno o ponteiro do vetor alocado, seguindo o protótipo:

   float reverso (int n, float *v)*

   Faça uma função main para testar sua função. Na função main, não esqueça de liberar a memória alocada pela função auxiliar.

2. Escreva uma função que receba como parâmetros duas strings e um caractere separador. A função deve criar a string que representa a concatenação das duas strings de entrada, usando o caractere como separador. Por exemplo, se forem passadas as strings “ex” e “aluno”, e o caractere hífen ‘-‘, deve-se ter como valor de retorno a string “ex-aluno”. O protótipo da função deve ser:

   char concatena (char *s1, char *s2, char sep);*

3. Crie um algoritmo que leia o tamanho de uma matriz de números reais e faça a alocação da matriz como um vetor de ponteiros. Crie uma função separada para ler e exibir a matriz original. Ao final, calcule o somatório de cada linha da matriz em uma função e armazene o resultado em um vetor. Exiba o vetor de somatório na tela.

4. Modifique o exercício do campo minado da AP1. Dessa vez, pergunte ao usuário o tamanho do campo minado e aloque uma estrutura do tamanho desejado. Realize a leitura, os cálculos e a exibição da matriz em funções separadas.

5. O uso de struct em C permite que tipos de dados complexos sejam gerados a partir de tipos básicos. Considere um tipo que representa um funcionário de uma empresa, definido pela estrutura a seguir:

```
typedef struct funcionario Funcionario;
struct funcionario {
  char nome[81]; /*nome do f u n c i o n á r i o */
  float valor_hora; /*v a l o r da hora de t r a b a l h o em Re ais */
  int horas_mes; /*horas t r a b a l h a d a s em um mês */
};
```

   Escreva um algoritmo que receba o número de funcionários a serem cadastrados e faça a alocação de um vetor do tipo Funcionario. Crie uma função que realize o cadastro de todos os funcionários. Além disso, crie uma função que realiza a busca de informações dos funcionários da empresa. O escopo da função deve ser:

   Funcionario busca (int n, Funcionario **v, char *nome);*

O programa deve permitir que o usuário procure os dados de funcionários através do nome, até que o usuário digite ‘0’ como nome. Se, numa determinada busca, o usuário for encontrado, os dados devem ser exibidos na tela. Se o usuário não for encontrado, o usuário deve ser avisado de que o funcionário não está cadastrado na empresa.

6. A função *malloc* deve receber o tamanho de um vetor. Mas, caso esse vetor seja insuficiente, a linguagem C oferece um mecanismo para realocar um vetor dinamicamente, com o uso da função *realloc* (também da biblioteca stdlib.h>). Dado um vetor vet, já alocado anteriormente, utiliza-se:

```
...
vet = (float *) realloc(vet, m*sizeof (float ));
..
```

A partir da realocação, “vet” passa a apontar para uma área de memória contígua suficiente para armazenar “m” valores reais. Conceitualmente, um realloc é equivalente a uma alocação, seguida de uma cópia de valores para a nova região da memória, seguida de uma liberação de memória da região antiga dos dados. Essas operações de realocação com cópia podem ser caras computacionalmente, então, usualmente, algumas otimizações são adotadas internamente nos computadores.

Com o uso dessa realocação, podemos criar **vetores dinâmicos**. Para demonstrar a ideia, crie um algoritmo que leia números positivos do usuário (até que ele digite um número negativo) e vá armazenando os valores lidos em um vetor. Se necessário, faça *realloc* do vetor, na medida em que mais posições forem sendo necessárias. Ao final, exiba o somatório acumulado dos valores do vetor.

OBSERVAÇÃO: o código deve ter uma constante que representa o passo de alocação/realocação do vetor. Por exemplo, se esse parâmetro for igual a 5, você alocará inicialmente um vetor de 5 posições e, caso sejam necessárias mais posições, você realocará o vetor com mais 5 posições, e assim sucessivamente (se o usuário for digitando mais e mais valores, já que você não sabe qual será o tamanho final que essa estrutura tem que ter).

7. Faça um algoritmo que pergunte ao usuário a quantidade de valores numéricos a serem digitados e aloque na HEAP um vetor com o tamanho especificado. Crie uma função que realiza a leitura dos números do vetor e outra função que exibe os números do vetor. Após isso, crie uma função que cria um vetor que contém apenas os números pares digitados pelo usuário. Ao final, exiba esses números pares.

   LEMBRE-SE: trate no código a situação de nenhum valor digitado ser par; para gerar o vetor de números pares, use vetores dinâmicos (a estratégia de *realloc* descrita no exercício anterior).