DIRETÓRIO ACADÊMICO DE COMPUTAÇÃO

Lista de exercícios Curso de C

2025.1

1) Considere a seguinte sequência numérica: 0, 1, 2, 3, 6, 11, 20, 37, 68, 125, 230, 423... Dado um valor inteiro n, crie uma função RECURSIVA que retorne o enésimo termo dessa sequência.

   Protótipo da função: int funcao_rec (int n)

   Exemplo de entrada: Saída esperada:

   n = 5 6

2) Escreva um programa em C para deletar um elemento na posição desejada de um array.

   Exemplo de entrada:

   Insira o tamanho do array: 5

   Insira 5 elementos:

   elemento- 0: 1

   elemento- 1: 2

   elemento- 2: 3

   elemento- 3: 4

   elemento- 4: 5

   Insira a posição onde excluir: 3

   Saída esperada: Array após a remoção: 1 2 3 5

3) O gerente de um museu deseja ter um circuito de câmeras de forma a monitorar pontos estratégicos, e você foi contratado para programar esse circuito. Assim, você deve criar um programa em C, que indique em que pontos essas câmeras devem ser instaladas.

   Além disso, o gerente do museu deseja reduzir o custo com a compra de equipamentos, portanto, a sua estratégia deve minimizar o número de câmeras utilizadas. Para o desenvolvimento do seu programa, o gerente irá fornecer um mapa do museu (Figura 1) - uma matriz $Mn \times n$, tal que, se Mij = 1, então há um corredor entre os pontos i e j, e Mij = 0 caso contrário.

   Protótipo da função: void printf_recursivo (int n_pontos, int pontos []);

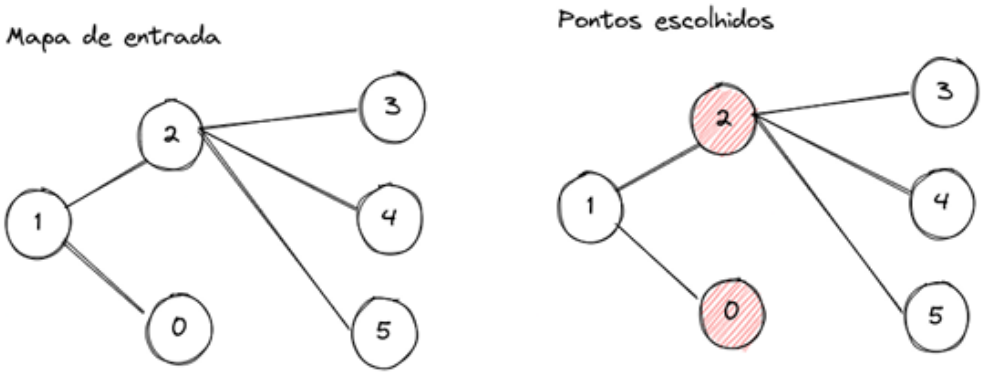

Figura 1: mapa do museu e pontos escolhidos para instalação das câmeras

Exemplo de entrada:

int M [6][6] = {{0, 1, 0, 0, 0, 0}, {0, 0, 1, 0, 0, 0}, {0, 0, 0, 1, 1, 1}, {0, 0, 1, 0, 0, 0}, {0, 0, 1, 0, 0, 0}, {0, 0, 1, 0, 0, 0}};

saída esperada:

Para monitorar todos os pontos, deve-se utilizar 2 câmeras, nos pontos 0 e 2.

4) Escreva um programa em C para encontrar a soma dos dígitos de um número utilizando recursão.

   Exemplo de entrada: 25

   Saída esperada: A soma dos dígitos de 25 = 7

5) Imagine que você é um programador de um sistema de inventário de uma grande loja. Você precisar criar uma função em C que organize o estoque de acordo com as posições dos produtos na prateleira. No estoque da loja, nem todos os produtos estão presentes e os que não estão têm -1 em seu lugar.

   Implemente uma função em C chamada organizar_estoque (arr, n) que recebe um array “arr” de tamanho “n”. O array “arr” contém os números dos produtos da loja onde “arr[i]” representa o produto “i”. Nem todos os produtos estão presentes no estoque, portanto, alguns elementos podem ter o valor -1. A função deve reorganizar, e mostrar na tela, o array de forma que arr[i] == i, se o produto” i” estiver presente no estoque, e “-1“caso contrário.

   Exemplo de entrada:
   arr1 = (-1, -1, 6, 1, 9, 3, 2, -1, 4, -1)
   arr2 = (19,7,0,3,18,15,12,6,1,8,11,10,9,5,13,16,2,14,17,4)
   Saída esperada:
   [-1, 1, 2, 3, 4, -1, 6, -1, -1, 9]
   [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]

6) Implemente uma função recursiva em C que:
   a) encontre e retorne o elemento máximo em um array de inteiros. A função deve receber o array e seu tamanho como parâmetros. A função deve ser implementada sem utilizar

estruturas de repetição (loops) e sem utilizar variáveis globais ou estáticas.
Assinatura da função: int find_max (int a [], int n)
Exemplo de entrada: Saída esperada:
int a [] = {5,8,2,1,4,7};
Int n = 6; Elemento máximo: 8

b) faça o mesmo para encontrar o elemento mínimo: int find_min (int a [], int n);

7) Suponha que você receba um array de inteiros nums, em uma etapa, remova todos os elementos nums[i] onde nums [i -1] < nums[i] para todo 0 < i < | nums | (máximo de 100). Crie um programa em C que mostre o número de etapas executadas até que nums se torne um array não-crescente. Além disso, seu programa deve imprimir o estado do array após cada etapa
Exemplo de Entrada: nums = [5,3,4,4,7,3,6,11,8,5,11]
Saída esperada:
Etapa 1: 5 3 4 3 8 5
Etapa 2: 5 3 3 5
Etapa 3: 5 3 3
Número total de etapas: 3

8) Implemente uma função recursiva em C que calcule e retorne a soma de todos os elementos de um array de inteiros. A função deve receber o array (0 < |array| <= 100) e o seu tamanho. A função deve ser implementada sem utilizar estruturas de repetição (loops) e sem utilizar variáveis globais ou estáticas. A assinatura da função deve ser: int findsum (int a [], int n);
Exemplo de entrada: int arr [] = {-1,-2,-3,-4,-5};
Exemplo de saída: Soma: -15

9) Crie uma função em C que receba três argumentos: um array preços[i] (representa o preço de uma ação no i-ésimo dia), um array bônus[i] (representa um bônus de venda que você recebe no i-ésimo dia), além do preço da ação, e um inteiro n: n <= 100 que representa o número de dias.

   Você deseja maximizar seu lucro escolhendo um único dia para comprar uma ação e escolher um dia diferente no futuro para vender essa ação, retorne o lucro máximo que você pode obter com essa transação. Se você não conseguir nenhum lucro, retorne 0. Você deve implementar a seguinte função, e imprimir na tela o dia da compra e o dia da venda: int maxProfit (int preços [], int bonus[], Int n)

   Exemplo de entrada: int preços [] = {7,1,5,3,6,4};
   Int bônus [] = {0,0,1,0,2,0);

   Exemplo de saída: Dia de compra: 1
   Dia de venda: 4
   O lucro máximo é: 7

10) Escreva uma função recursiva equivalente ao seguinte algoritmo:

```
Int main () {
     Int x,y = 1, n;
     Scanf (“%d”, &n);
     For (x = 1; x <= n; x++)
          Y = y*x;
            Printf (“%d”, y);
            Return 0;
}
```

11) Dado um valor inteiro (ex: 12345, 0123), implemente as seguintes funções em C:

A) Função que contabiliza a quantidade de dígitos de num de maneira recursiva: Int Count_digit_recursive (int num)
B) Função que mostra o número num com seus dígitos invertidos: void inverter (int num)
C) Função que retorne 1 caso num seja palíndromo e 0 caso contrário: int is_palindrome (int num)

12) Escreva uma função recursiva equivalente ao algoritmo:

```
Int function (int n) {
    If (n == 0) return 0;
    If (n == 1) return 1;
    Int a = 0;
    Int b = 1;
    Int c = 0;
    For (int i = 2; i <= n; i++) {
        c = b + 1;
        a = b;
        b = c;
    }
    Return c;
}
```

13) Dado um valor inteiro em que seus dígitos estão em uma sequência não decrescente (ex: 12345,0123). Implemente as funções:

a) Função que soma os dígitos de num de maneira recursiva: Int sum_digits_recursive (int num)
b) Função que mostra o número num com seus dígitos invertidos: void inverter (int num)
c) Função para retornar o digito faltante da sequência: int missing_digit (int num)

d) Função que conta o número de divisores de num de maneira recursiva, iniciando a contagem com divisor = 1: int Count_divisors_recursive (int num, int divisor)

14) Crie uma função que retorne 1 se o número inteiro fornecido for simétrico em relação ao número de seus dígitos (o número lido da esquerda para a direita é igual ao número lido da direta para a esquerda), caso contrário a função deve retornar 0, seu traço é: int is_symmetric (int x)

15) Implemente uma função recursiva em C (soma_diagonal_secundaria) que retorne a soma da diagonal secundária de uma matriz quadrada, sem usar laços. traço da função: int soma_diagonal_secundaria (int n, const int matriz[n][n]);

16) Escreva uma função chamada ordenarCores que recebe um vetor contendo apenas os números 0,1 e 2. A função deve reordenar o vetor de forma que todos os 0s venham primeiro, seguidos por todos os 1s, e finalmente, por todos os 2s. SEM usar funções de ordenação prontas. traço da função: void ordenarCores (int nums [], int numsSize);

17) Escreva um programa em C para dividir um vetor a cada enésimo elemento, tal que n seja lido do teclado.

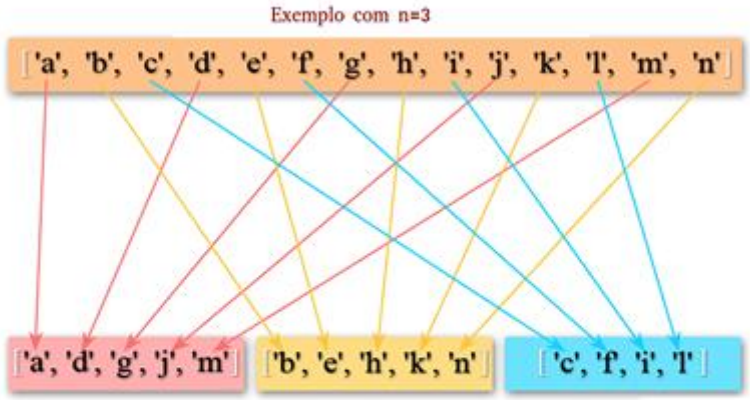

Figura 1: Exemplo com n=3.

Ex. de entrada: {’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’}
Saída: {{’a’, ’d’, ’g’, ’j’, ’m’}, {’b’, ’e’, ’h’, ’k’, ’n’}, {’c’, ’f’, ’i’, ’l’}}

18) Escreva uma função em C que receba um vetor de inteiros como argumento e determine se todos esses inteiros são diferentes uns dos outros.

Ex. 1 de entrada: Saída esperada:
[1,5,7,9] True
Ex. 2 de entrada: Saída esperada:
[2,4,5,5,7,9] False

19) Escreva um programa em C para receber um valor N de entrada e “desenha” na tela um triângulo retângulo com n linhas usando asteriscos.

20) Encontre o número faltante em um array, dado um array de n-1 inteiros distintos em que os elementos estão no intervalo de 1 a n,

crie um programa em C que encontre o número que falta nele, tal que: existe somente um número faltando, 2 <= n <= 1000, os elementos são distintos e estão no intervalo de 1 a n.

Assinatura da função: int missing_number (int arr [], int n);

21) Imagine que você é um investidor na bolsa de valores e tem X reais em caixa, disponíveis para o investimento. Na sua carteira de ações, você possui N ações disponíveis para o investimento. Cada ação i possui um retorno financeiro li e um custo ci, caso você decida investir nela.
Considere que ou você investe na ação ou não investe. Não é permitido investir uma porcentagem da ação. Por exemplo, se uma ação 1 possui l1 = 10 e c1 = 5, e você decide investir nela, o seu retorno financeiro com essa ação será de 10 reais e o custo será de 5 reais.

Crie um programa em C que, dado X reais em caixa e N ações com seus respectivos custos e retornos, encontre o melhor investimento possível, maximizando o retorno financeiro total. Deverão ser mostrados em arquivo, quais ações foram escolhidas e o valor do retorno financeiro total atingido.

Obs.: Lembre que você tem apenas X reais para investir nas ações, portanto, o custo total do investimento não pode ultrapassar X.
Exemplo de entrada:
X = 77, n = 7
Lucros [7] = {70, 20, 29, 37, 7, 5, 10}
Custos [7] = {31, 21, 20, 19, 4, 3, 6}
Possível saída (em arquivo):
As ações escolhidas foram: 0, 3, 4, 5 e 6
Lucro total = 129.00

22) Você foi contratado para gerenciar uma fábrica, na qual possui diversas máquinas que automatizam a produção. Sua função consiste em, estrategicamente, atribuir n tarefas a n máquinas, de modo que toda tarefa seja atribuída, e que cada máquina realize uma e somente uma tarefa, e que além disso o custo total desta atribuição seja mínimo.

Para tal, você deve criar um programa em C, que determine a alocação de máquinas a tarefas, de modo a minimizar o custo total da alocação, e de modo que nenhuma máquina execute mais do que 1 tarefa. Um exemplo de custo de cada máquina para a execução de cada tarefa, pode ser visualizada na Figura 1.

Custos de Execução das Tarefas x Máquinas ($)					
	Tarefa 1	*Tarefa 2*	*Tarefa 3*	*Tarefa 4*	*Tarefa 5*
Maq. 1	23,00	30,00	27,00	20,00	21,00
Maq. 2	21,00	29,00	25,00	17,00	19,00
Maq. 3	28,00	36,00	32,00	25,00	25,00
Maq. 4	25,00	34,00	29,00	20,00	22,00
Maq. 5	24,00	33,00	28,00	21,00	23,00
Maq. 6	27,00	38,00	32,00	25,00	23,00

Figura 1: Custo Tarefa x Máquina

23) Em uma civilização antiga, números eram representados por sete símbolos diferentes: A = 1, E = 5, I = 10, O = 50, U = 100, @ = 500, # = 1000.
Por exemplo, 2 é escrito como AA, que é a soma de dois uns; 12 é IAA (I + A + A), esses números são normalmente escritos do maior para o menor e da esquerda para a direita, porém, o número quatro não é escrito como AAAA, mas sim com AE, já que o A está antes do cinco, subtrai seu valor (resultando em 4), mesma lógica para o nove (AI).
Os seis casos em que a subtração é usada são:

- A pode ser colocado antes de E (5) e I (10).
- I pode ser colocado antes de O (50) e U (100).
- U pode ser colocado antes de @ (500) e # (1000).

Dado um numeral str, crie uma função int ancientToInt (char*); que converta a string recebida para um número inteiro
Exemplo de entrada: string = “AAA”
Saída esperada: 3

24) A startup “techBooks” possui uma base de dados de livros, onde cada livro é representado por uma estrutura (struct) contendo o título do livro (máximo de 100 caracteres), o autor (máximo de 50 caracteres) e o ano de publicação.
A startup está migrando seus dados para um novo sistema e precisa de um programa que carregue esses dados de um arquivo texto para a memória, realize algumas operações e, por fim, salve os dados modificados em um novo arquivo. Seu papel como programador é escrever uma função void salvarLivros(Livro* livros, int n, char* nomeArquivo) que receba o array de livros (alocado dinamicamente), o número de livros e o nome do arquivo. Esta função deve abrir o arquivo e salvar os livros do array no arquivo (um livro por linha no formato: título; autor; ano)

25) Imagine que você tem um conjunto de bolsas, cada uma contendo diferentes itens. Cada item possui um número escrito nele, que pode ser 1,0 ou -1. Seu objetivo é escolher exatamente k itens de qualquer uma dessas bolsas para maximizar a soma dos números escritos nos itens selecionados.
Escreva uma função em C que, dado um conjunto de bolsas e um número k, retorne a soma máxima possível que pode ser obtida escolhendo exatamente k itens e quantos itens de cada número (1,0,-1) foram escolhidos para alcançar essa soma.

As bolsas devem ser lidas de um arquivo de texto. Com o formato:
<num bolsas>
<num_uns_1> <num_zeros_1> <num_negativos_1>
<num_uns_2> <num_zeros_2> <num_negativos_2>
...
<num_uns_n> <num_zeros_n> <num_negativos_n>

A função deve ser: int* k_itens_com_max_soma (Bolsa* bolsas, int num_bolsas, int k);

Ex de entrada (arquivo bolsas.txt) e int k = 6:
3
3 2 0
1 1 3
1 0 1

Saída esperada:
Soma máxima: 5 num_uns: 5
Num_zeros: 1 num_negativos: 6

26) Você está coordenando a distribuição de recursos em um evento comunitário. Cada participante tem uma necessidade mínima (necessidades[i]) e você dispõe de recursos com valores variados (recursos[j]). Se um recurso j atende ou supera a necessidade de um participante i ele pode ser alocado para este participante, satisfazendo sua necessidade. Seu objetivo é alocar os recursos para maximizar o número de participantes satisfeitos. Além disso, você deve registrar em um arquivo quais recurso foram alocados para quais participantes. O arquivo deve conter em cada linha, um participante e recurso alocado para ele no formato: “participante i

recebeu o recurso j", no final deve ser mostrado o número de participantes satisfeitos.
O traço da função é: void alocar_recursos (struct DistribuicaoRecursos dados, const char* nomeArquivo);

Exemplo de entrada: int necessidade [] = {1,2};
Int recursos [] = {1,2,3};
DistribuicaoRecursos dados = {necessidades, recursos, 2, 3};
Saída esperada:
Participante 0 recebeu recurso 1
Participante 1 recebeu recurso 2
2

27) Escreva um programa em C para contar o número de palavras e vogais em um arquivo.

Arquivo de entrada:	Saída:
prova 1	
LP	3 palavras e 2 vogais

28) Crie um programa em C para o cadastro de pessoas a partir do teclado em um vetor dinâmico (Pessoa* agenda), inicialmente de tamanho igual a zero. Cada vez que uma nova pessoa for introduzida, o bloco de memória apontado por agenda é aumentado pelo tamanho de uma Pessoa. Para encerrar o cadastro e sair do programa, o usuário deverá entrar com um nº de telefone negativo.

```
typedef struct {
char nome [100];
int telefone;
} Pessoa;
```

Dica: utilize as funções void* malloc(unsigned size) e void* realloc (void* ptr, unsigned size)

29) Crie uma função em C que implemente a rotação à esquerda de um vetor em N positions posições. A função deve ter a seguinte assinatura:

void rotateLeft (int *arr, int size, int n_positions);

- *arr: é o ponteiro para o início do vetor de inteiros a ser rotacionado.
- size: é o número total de elementos no vetor.
- n positions: é o número de posições para as quais o vetor deve ser rotacionado à esquerda.

Ex. de entrada: Array inicial: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
Número de posições para rotacionar: 3
Saída esperada: Array após a rotação: [40, 50, 60, 70, 80, 90, 100, 10, 20, 30]

30) Crie uma função em C para realizar a multiplicação de duas matrizes. Ambas as matrizes de entrada e a matriz resultante devem ser alocadas dinamicamente usando ponteiros para ponteiros (“int**”). A função deve verificar se a multiplicação é possível e, em caso negativo, retornar “NULL”.

Traço da função: int** multiplyMatrices (int** mat1, int rows1, int cols1, int** mat2, int rows2, int cols2);

- mat1: Ponteiro para a primeira matriz (alocada dinamicamente).
- rows1: Número de linhas da primeira matriz.
- cols1: Número de colunas da primeira matriz.
- mat2: Ponteiro para a segunda matriz (alocada dinamicamente).
- rows2: Número de linhas da segunda matriz.
- cols2: Número de colunas da segunda matriz.

Valor de Retorno:

• Um ponteiro para a nova matriz resultante (alocada dinamicamente), se a multiplicação for bem-sucedida.
• NULL, se a multiplicação não for possível (o número de colunas da primeira matriz não for igual ao número de linhas da segunda matriz).

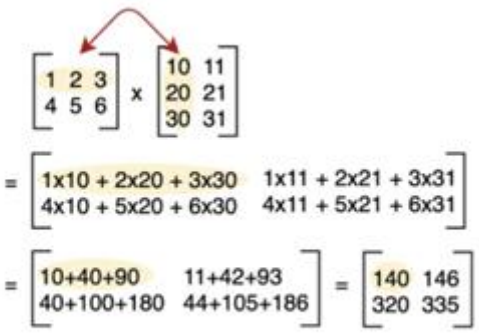

Figura 1: Exemplo de multiplicação

31) Você precisa desenvolver um sistema em C para gerenciar dados de alunos, que incluem nome, matrícula e duas notas.Estes dados devem ser persistidos em um arquivo .txt para poderem ser salvos.
Defina uma struct chamada Aluno para armazenar as seguintes informações: nome, matrícula, nota1 e nota2.

Em seguida, implemente a seguinte função para salvar Alunos em Arquivo de Texto: void salvarAlunos (Aluno alunos [], int numAlunos, const char* nomeArquivo);

• alunos []: é o array de Alunos a ser salvo.
• numAlunos: é o número de alunos no array.
• nomeArquivo: é o nome do arquivo de texto onde os dados serão salvos (ex: "dadosalunos.txt").

32) Crie uma função em C que, dado um array de números inteiros, mova todos os zeros para o final do array. A ordem relativa de todos os outros elementos (não-zeros) deve ser mantida. A função deve ter a seguinte assinatura: void pushZerosToEnd (int *arr, int size);

•arr []: é o array de inteiros a ser modificado.

•size: é o número total de elementos no array.

Ex. de entrada: Array inicial = [1, 9,8, 4,0, 0,2, 7,0, 6,0]

Saída esperada: Array após a operação = [1,9, 8,4, 2,7, 6,0, 0,0, 0]

33) Dado um vetor real X com N elementos que é apresentado como Uma possível solução de um sistema de equações lineares Ax=B, onde A é uma matriz real m×n e B é um vetor real de m elementos (lado direito das equações). Crie uma função em C que verifique se o vetor X é realmente uma solução válida para o sistema dado.

•A comparação entre números de ponto flutuante deve ser feita usando uma tolerância (epsilon) devido a imprecisões.

•Todas as matrizes e vetores devem ser tratados como alocados dinamicamente.

int verifySolution (double** A, int m, int n, double* X, double* B, double tolerance);

•A: Ponteiro para a matriz de coeficientes A (alocada dinamicamente, m linhas, n colunas).

•m: Número de linhas da matriz A (e elementos do vetor B).

•n: Número de colunas da matriz A (e elementos do vetor X).

•X: Ponteiro para o vetor candidato à solução X (alocado dinamicamente).

•B: Ponteiro para o vetor do lado direito das equações B (alocado dinamicamente).

•tolerance: Um pequeno valor real para a comparação de números de ponto flutuante (ex: 0.0001).

ValordeRetorno:

•1(ou verdadeiro), se o vetor X for uma solução válida para o sistema Ax=B.

•0(ou falso), caso contrário.

Ex. de entrada:

$$A = \begin{bmatrix} 2.0 & -2.0 & 1.0 \\ 1.0 & 3.0 & -2.0 \\ 3.0 & -1.0 & -1.0 \end{bmatrix} \qquad X = \begin{bmatrix} -7/5 \\ -2 \\ -21/5 \end{bmatrix} \qquad B = \begin{bmatrix} -3.0 \\ 1.0 \\ 2.0 \end{bmatrix}$$

Tolerância: `0.0001`

Saída esperada: Valor de retorno da função: 1 (Verdadeiro)

34) Escreva um programa em C para substituir uma linha específica por outro texto em um arquivo. Suponha que o conteúdo do arquivo test.txt seja:

test line 1
test line 2
test line 3
test line 4

DadosdeTeste:

• Nome do arquivo: test.txt
• Novo conteúdo da linha: Sim, eu sou o novo texto emvezdalinha2
• Número da linha a substituir: 2

Saída Esperada no Terminal: Substituição realizada com sucesso!

35) Desenvolva um programa em C para gerenciar um catálogo de carros. O programa deve ser capaz de armazenar informações como placa e modelo, utilizando uma estrutura de dados chamada carro. As informações de cada carro devem ser inseridas

dinamizamente na memória e gerenciadas por um array de structs de tamanho n.

a) Crie uma struct Carro.
b) Implemente a função char* get_modelo (Carro*, char*) que busca um carro por meio de sua placa, e retorna o seu modelo.

36) Implemente uma função em C int maxProfit (int*, int) que recebe um array de preços, em que preços[i] representa o preço de uma determinada ação no i-ésimo dia. Você deseja maximizar seu lucro escolhendo um único dia para comprar uma ação e escolher um dia diferente no futuro para vender essa ação. A função deve retornar o lucro máximo que você pode obter com essa transação e mostrar na tela os dias escolhidos. Se não for possível obter lucro retorne 0.
Exemplo de entrada: precos = [7,1,5,3,6,4]
Saída esperada: 5
Compre no dia 1 (preço = 1) e venda no dia 4 (preço = 6), lucro = 6 -1 = 5

37) Escreva um programa em C para comparar duas strings sem usar funções de biblioteca de strings, as duas strings deverão ser lidas no teclado para variáveis alocadas dinamicamente. O tamanho pode ser definido pelo programador previamente.
A função deve retornar 1 se as strings forem iguais, 0 se não.
Traço da função: int compare (char* str1, char* str2);

38) Crie um programa em C que recebe duas Structs que representam eventos que aconteceram no mesmo dia, event_1 e event_2, onde:

```
Struct {
  Float start_period;
  Float end_period;
```

```
} Event;
```

Um conflito ocorre quando dois eventos têm alguma intersecção não vazia (ou seja, algum momento é comum a ambos os eventos). O código deve retornar 1 (true) se houver conflito entre os dois eventos, caso contrário retorne 0 (false).
Exemplo de entrada: event_1 = {1.15, 2.0}, event_2 = {2.0, 3.0}
Saída esperada: 1 (verdadeiro)

39) Você está desenvolvendo um software para a manipulação e análise de matrizes n X m em um sistema de processamento de dados. Nesse sistema, é necessário implementar um algoritmo que permita realizar diversas operações em matrizes representadas como vetores. Para atender a esses requisitos, você precisa implementar as seguintes funções abaixo:

a) Função que retorna o tipo da matriz, -1 se for assimétrica e 1 se for simétrica (para todos os termos da matriz, a[i][j] = a[j][i]): Int tipoDeMatriz (int* vet, int n, int m);
b) Função que retorna um vetor com os elementos da linha l da matriz: int* linhaDaMatriz (int* vet, int n, int m, int l);

40) Considere a seguinte definição de uma estrutura chamada Student que armazena informações sobre um estudante:

```
Typedef Struct {
  Char nome [50];
  Int idade;
  Float* notas;
  Int numNotas;
} Estudante;
```

Escreva uma função chamada Create Student que recebe como argumento o nome e a idade de um estudante, cria uma instância de Student, aloca memória para o array de notas com base em um

valor fornecido pelo usuário e inicializa numNotas como zero. Em seguida, implemente uma função chamada addGrade que recebe um ponteiro para uma instancia Student, a nota do estudante e atualiza o array de notas. Certifique-se de realocar a memória do array de notas para acomodar a nova nota e atualizar o valor de numNotas.