1)Considere o tipo Candidato abaixo:

struct candidato

{ int inscrição;

char nome[51];

int idade;

float nota;

};

typedef struct candidato Candidato;

Escreva as funções:

1. criaCandidato: que recebe uma inscrição, um nome, uma idade e uma nota, e cria, com alocação dinâmica, um novo candidato, preenchendo seus campos. A função retorna o endereço do novo candidato criado, ou, caso não tenha sido possível criar o candidato, a função retorna NULL;
2. exibeTodosCandidatos: a função recebe um vetor de ponteiros para candidatos e o número de candidatos, exibindo os dados de todos os candidatos;
3. buscaInscricao: a função recebe um vetor de ponteiros para candidatos, o número de candidatos e uma inscrição. A função retorna o endereço do candidato, se ele for encontrado, ou NULL, caso contrário.

Use a função main abaixo para testar as funções desenvolvidas.

int main (void) {

Candidato \*vpcandidato[8]; int i;

Candidato \*pont;

vpcandidato[0]= criaCandidato(444,"Luiz",34, 7.8f);

vpcandidato[1]= criaCandidato(111,"Rita",56, 8.8f);

vpcandidato[2]= criaCandidato(999,"Rute",32, 7.1f);

vpcandidato[3]= criaCandidato(555,"Dina",27, 6.4f);

vpcandidato[4]= criaCandidato(777,"Lana",35, 5.3f);

vpcandidato[5]= criaCandidato(666,"Tais",29, 9.8f);

vpcandidato[6]= criaCandidato(222,"Cris",31, 7.2f);

vpcandidato[7]= criaCandidato(333,"Vera",44, 5.4f);

exibeTodosCandidatos(vpcandidato, 8);

/\* inclua aqui a parte referente ao teste da função buscaInscricao: leitura de uma inscrição, chamada da função e exibição dos dados do candidato, se encontrado \*/

for (i=0; i<8; i++)

free(vpcandidato[i]);

return 0;

}

2) Considere os tipos definidos abaixo para representar um cadastro de alunos de uma turma:

#define MAX 100 struct aluno {

int mat; /\* matricula \*/

char nome[81]; /\* nome \*/

float nota; /\* nota final \*/

};

typedef struct aluno Aluno;

struct turma {

int n; /\* número de alunos máximo da turma alocados \*/

Aluno\* vet[MAX]; /\* vetor de ponteiros para alunos \*/

};

typedef struct turma Turma;

1. Implemente uma função para criar uma nova turma, inicialmente vazia, alocada dinamicamente: Deve inicializar todas as posições com NULL

Turma\* cria\_turma (void);

1. Implemente uma função para adicionar um novo aluno na turma, dadas suas informações: Oaluno deve ser incluído no primeiro "buraco" encontrado

int adiciona\_aluno (Turma\* t, int mat, char\* nome, float nota);

retorna 1 caso a inclusão seja bem sucedida,

0 caso contrário ( não tem espaço para incluir)

1. Implemente uma função para retirar um aluno da turma, dado sua matrícula. No local da retirada fica um "buraco" (c/ NULL) e os dados do aluno retirado são exibidos.

int retira\_aluno (Turma\* t, int mat);

retorna 1 caso a retirada seja bem sucedida,

0 caso contrário ( não encontrou o aluno)

1. Implemente uma função para calcular a media das notas dos alunos de uma turma:

float notamedia (Turma\* t);

1. Implemente uma função para exibir os dados dos alunos de uma turma: Se a turma estiver vazia, enviar mensagem adequada.

void exibeturma (Turma\* t);

(f) Implemente uma função para liberar a memória alocada para uma turma, e seus respectivos alunos:

float libera\_turma (Turma\* t);

3) Considere o tipo estruturado Endereco abaixo:

struct endereço

{ char rua[31];

int numero;

char complemento[15];

char bairro[22];

};

typedef struct endereço Endereco;

3.a) Escreva a função **leDadosDeEndereco** que recebe uma (ponteiro para ) estrutura Endereço e a preenche com os dados lidos do teclado

Para os próximos itens da questão 3, considere, para representar um imóvel do cadastro de uma imobiliária, o tipo estruturado Imovel a seguir:

struct imovel

{ char tipo[15]; /\* casa, apto, casa de vila, outro \*/

Endereco ender;

float valor;

char proprietário[60];

}

typedef struct imovel Imovel;

3.b) Escreva a função **leDadosImóvel** que aloca dinamicamente um imóvel, retornando -o preenchido O tipo, valor e proprietário, são recebidos pela função. Os dados do endereço são lidos pela função 3a)

3.c) Escreva a função **contaNoBairro**, que recebe o número de imóveis da imobiliária, o vetor de ponteiros de imóveis e o nome de um bairro, e, retorna o o número de imóveis nesse bairro.

3.d) Escreva a função **copiaNoBairro**, que recebe o número de imóveis da imobiliária, o vetor de ponteiro de imóveis e o nome de um bairro, e cria um novo vetor de ponteiros que aponta para os imóveis deste bairro ( não cria cópias dos imóveis, aponta para os imóveis originais)

3.e) Escreva a função **mostraImóveis**, que recebe o número de imóveis, um vetor de ponteiros de imóveis e exibe os dados dos imóveis

4) O cadastro dos funcionários de uma certa empresa de transportes (a TRAN) utiliza um vetor de ponteiros para o tipo estruturado Funcionario abaixo que representa um funcionário dessa empresa.

struct funcionario

{ char motorista[51];

int qtDeMultas; /\* quantidade de multas desse motorista \*/

};

typedef struct funcionario Funcionario;

4.a) Escreva a função criaFuncionario, que recebe o nome e o a quantidade DE MULTAS DE um novo funcionário e cria um novo funcionário com esses dados, utilizando alocação dinâmica. Caso não seja possível criar tal funcionário, a função retorna NULL.

4.b) Escreva em C a função **contaPioresMotoristas**, que recebe o número de funcionários da empresa, o vetor de de ponteiros para funcionários, e retorna o número de motoristas com quantidade de multas >10.

5) Defina um tipo adequado para representar um acervo de CD´s, sabendo que cada CD deve ser identificado por um número além das informações usuais.

CD:

Número

Nome

Banda

Gravadora

Ano de gravação

Número de faixas

relação de músicas ( com tamanho exato )

Musica:

Faixa

Nome

Tempo

Compositor

Obs: Nenhuma string deve ter tamanho maior que a qtdade de caracteres necessárias para armazená-la

Faça uma função que leia os dados de um CD, criando-o dinamicamente e retornando-o

faça uma função que preencha o acervo como vetor de ponteiros. Pergunta quantos CDs o usuário deseja armazenar e, ativando a funçaõ anterior, prenche o acervo

Faça uma função recursiva que receba o acervo de CD´s de uma pessoa,armazenado em um vetor de ponteiros para estruturas, e exiba o número de cada um dos CD´s da década de 60.

Faça uma função que receba o acervo de CD´s de uma pessoa, armazenado em um vetor de ponteiros para estruturas, e exiba os CDs com maior número de trilhas

Faça uma função que receba o acervo e o nome de um compositor, exibindo todos os Cds que tem pelo menos uma música deste compositor

Faça uma função que "destrua" a estrutura do acervo

Faça um programa que pergunte ao usuário o tamanho de seu acervo, crie um vetor de ponteiros para os CDs e preencha-o utilizando a função anterior e, a seguir, exiba os CDs dos anos 60, os CDs com maior número de faixas, pergunte o nome de um compositor, mostrando os Cds que possuem pelo menos uma múscia deste compositor e depois "destrua a estrutura"

Faça uma função que receba o acervo de CD´s e retorne 1, caso todos os CDs tenham a mesma quantidade de faixas ou 0 caso contrário

Faça uma função que receba o acervo e o nome de um compositor, exibindo todos os Cds que tem pelo menos uma música deste compositor==> **MODIFICAÇÂO: exibir apenas o nome das músicas do compositor**