Exercício 1:
Escreva um programa fazendo o uso de struct's. Você deverá criar uma struct chamada Ponto, contendo apenas a posição x e y (inteiros) do ponto. Declare 2 pontos, leia a posição (coordenadas x e y) de cada um e calcule a distância entre eles. Apresente no final a distância entre os dois pontos.

Exercício 2:
Seja a seguinte struct que é utilizada para descrever os produtos que estão no estoque de uma loja :

```
typedef struct sProduto
{
        char nome[30]; /* Nome do produto */
        int codigo;
        /* Codigo do produto */
        double preco;
        /* Preco do produto */
} Produto;
```

a) Escreva uma instrução que declare um vetor de Produtos com 10 itens de produtos;

b) Atribua os valores "Pe de Moleque", 13205 e R$ 0,20 aos membros da posição 0 e os valores "Cocada Baiana", 15202 e R$ 0,50 aos membros da posição 1 da matriz anterior;

c) Faça as mudanças que forem necessárias para usar um ponteiro para Produto ao invés de um vetor de Produtos. Faça a alocação de memória de forma que se possa armazenar 10 produtos na área de memória apontada por este ponteiro e refaça as atribuições da letra b;

d) Escreva as instruções para imprimir os campos que foram atribuídos na letra c.

Exercício 3:
Escreva um programa que irá implementar uma agenda com o seguinte Menu:

```
AGENDA
==================
1 Insere contato
2 Remove contato
3 Busca nome
4 Imprime por iniciais
0 Sair
```

A estrutura contato deverá conter 3 strings: nome (20 posições), telefone residencial (12 posições) e telefone celular (12 posições). O programa deverá ser capaz de armazenar até 100 posições. Inicialmente, o campo nome em todos os contatos deverá estar vazio ( ), sendo esse o critério a ser utilizado para saber se uma posição está ou não preenchida.

A inserção e remoção de contatos deverá ser feita através do nome. O programa deverá checar se o nome já existe ao inserirmos (possibilitar a edição deste contato) e deverá fazer o mesmo teste antes de remover (neste caso, se não existir, imprimir mensagem de erro informando).
A busca será feita por nome e, se encontrar, irá informar o restante das informações do contato.
A opção para imprimir iniciais irá pedir ao usuário apenas a primeira letra do nome, imprimindo as informações de todos os contatos que sejam iniciados por esta letra.

Exercício 4:
Crie um programa para cadastro simples de alunos. O programa deverá suportar o cadastro de até 1000 alunos (vetor). Para cada aluno, deveremos armazenar o nome (char nome[50]), matricula (char mat[15]) e Coeficiente de rendimento (float CR). O programa possuirá o seguinte menu:
CADASTRO DE ALUNOS
==========
1 Insere aluno
2 Remove aluno
3 Imprime Lista Completa
4 Imprime média do CR
5 Imprime alunos por CR específico
6 Imprime alunos por inicial
0 Sair

A iniciar o programa, o vetor deverá conter o valor - 1 no campo CR de cada posição, indicando que a mesma esta vazia. Para inserir um novo aluno, deve-se verificar a primeira posição no vetor com CR negativo (menor do que zero).
A remoção será feita por matrícula. Deve-se informar a matrícula, buscar o aluno na lista e removê-lo (atribuir ao campo CR o valor -1). Se o aluno não for encontrado, deverá ser mostrada uma mensagem de erro (Aluno não encontrado). A impressão da lista completa deverá ter o seguinte formato:
Matricula CR Nome do aluno.

Imprima o campo CR sempre com o mesmo tamanho (utilize %4.1f por exemplo no printf). Recomendável dar uma pausa durante a impressão se a lista for muito grande para que todos os dados dos alunos possam ser lidos. Imprima a média do CR como visto no exemplo em sala. Já na impressão com CR específico, será perguntado o CR mínimo e máximo, seguido da impressão dos alunos que estiverem nesse intervalo fechado. Na impressão por inicial, deverá ser lido um caractere e imprimir os alunos cujo primeiro nome tenha esse caractere como inicial.