Considere as seguintes definições de registros e a variável opcionais:

```
#define  TAM     50

typedef struct {
    char placa[9];
    char modelo[TAM];
    char fabricante[TAM];
    int ano_fabricacao;
    int ano_modelo;
    char combustivel[TAM];
    char cor[TAM];
    int opcional[8];
    float preco_compra;
} CARRO;
```

```
char opcionais[][TAM]={
            {"4.portas"},
            {"cambio.automatico"},
            {"vidros.eletricos"},
            {"abs"},
            {"air.bag"},
            {"ar.condicionado"},
            {"banco.couro"},
            {"sensor.estacionamento"}};
```

```
typedef struct {
    char nome[TAM];
    char cpf[15];                  //999.999.999-99
    ENDERECO endereco;
    TELEFONE residencial;
    TELEFONE celular[5];
    int quantos_celulares;
    float renda_mensal;
} CLIENTE;
```

```
typedef struct {
    char rua[TAM];
    int numero;
    char bairro[TAM];
    char cidade[TAM];
    char estado[3];
    char cep[11];       //99.999-999
} ENDERECO;


typedef struct {
    char telefone[14];    //99 99999-9999
} TELEFONE;
```

```
struct DATA {
   int dia, mes, ano;
};

struct VENDA_CARRO {
    char placa_car[9];
    char cpf_cli[15];
    float preco_venda;
    struct DATA data_venda;
};
```

O campo “opcional” de CARRO é um vetor para indicar quais opcionais um carro tem. Por exemplo, o conjunto {1,0,0,0,1,0,0,1} armazenado no vetor indica os seguintes opcionais de um carro: "4.portas", "air.bag" e "sensor.estacionamento".

desenvolva um programa em C/C++ usando MÓDULOS e ARQUIVOS BINÁRIOS com as seguintes funcionalidades acessíveis através de MENUS de opção:

1. Menu Carro
    a. Inserir um carro no cadastro
    b. Excluir um carro do cadastro
    c. Listar todos carros disponíveis para venda
    d. Listar os carros disponíveis para venda que tem um determinado opcional

2. Menu Cliente
    a. Inserir um cliente no cadastro
    b. Listar todos clientes

3. Menu Venda
    a. Inserir uma venda
    b. Excluir uma venda
    c. Listar os carros vendidos de um determinado fabricante, em ordem crescente pelo modelo (as seguintes informações devem aparecer: modelo, placa, ano fabricação e nome cliente)
    d. Listar os carros vendidos de um determinado modelo, em ordem crescente pelo ano de fabricação (as seguintes informações devem aparecer: ano de fabricação, placa e nome cliente)
    e. Informar a quantidade de carros vendidos com o valor totalizado dos preços vendidos
    f. Informar o lucro total das vendas

4. Encerrar o programa

OBSERVAÇÕES:

- Para inserir um carro gere automaticamente as informações do carro. Cada carro tem uma placa única, então não pode ser inserido um carro com uma placa que já existe no arquivo.
- Na exclusão de um carro use a placa como pesquisa e o carro poderá ser excluído se não existir uma venda desse carro. A exclusão deve ser física, portando o registro CARRO não pode ser alterada sua estrutura.
- Carros disponíveis para venda são aqueles que ainda não foram vendidos.
- Todas as informações do carro devem aparecer em cada listagem tendo o "nome do campo" e a "informação".

- Para inserir um cliente gere automaticamente as informações do cliente. Cada cliente tem um cpf único, então não pode ser inserido um cliente com um cpf que já existe no arquivo.
- Todas as informações do cliente devem aparecer na listagem tendo o "nome do campo" e a "informação".

- Para inserir uma venda o carro e o cliente devem existir nos cadastros.
- Para excluir uma venda use a placa do carro como pesquisa. A exclusão deve ser física, portando o registro VENDA_CARRO não pode ser alterada sua estrutura.

- Utilize as rotinas da biblioteca "bibNCP.a" para gerar automaticamente informações para um carro ou um cliente. Consulte e use "unit.h" para saber mais sobre essas rotinas.
- O programa deverá rodar no Linux.