1) Crie uma classe *Carro* com os atributos **ano, marca, modelo, cor, itensDeSerie** (List do tipo *Item*) e **motor**. Crie também uma classe *Item* para representar os itens de série, com os atributos **descricao** e **valor**. Crie a classe *Motor* com os atributos **potencia** e **qtdCilindros**. Todos os atributos devem ser privados.

Adicione na classe *Carro* o método **adicionaItemDeSerie(Item item)** que deve adicionar um item de cada vez na lista de itensDeSerie.

Faça uma nova classe que tenha o método main e preencha todas as informações das classes criadas acima, depois disso imprima no seguinte formato:

> Descrição do carro: Volkswagen, Gol, 2000, Preto
>
> Itens de serie: Ar condicionado vidro elétrico trava elétrica
>
> Motor: 110 cv, 4 cilindros

2) Crie uma classe *Conta* (**numero, saldo e cliente**) e uma classe *Cliente* (**nome, cpf e rg**).
Faça com que ao ser criada uma conta, seja obrigatório informar o cliente que esta conta pertence.

Faça uma nova classe que tenha o método main e preencha todas as informações das classes criadas acima, depois disso imprima no seguinte formato:

> Dados da conta
>
> Número da conta: 00000002-1
>
> Saldo: 20,00
>
> Dados do cliente:
>
> Nome: José da silva
>
> CPF: 111.111.111-11
>
> RG: 44.444.444-4

3) 1,5pts Crie uma classe *Pessoa* com o atributo **nome** e com o método **imprimir**, esse método deve imprimir os dados da pessoa da seguinte forma:

```
Nome: Maria
```

Crie uma outra classe *PessoaFisica* que herde de *Pessoa*, essa classe deve ter o atributo **cpf** além de herdar o **nome** da *Pessoa*. Faça o mesmo para *PessoaJuridica* porém adicionando o atributo **cnpj** ao invés de cpf.

Faça com que o método **imprimir** seja alterado nas classes filhas, imprimindo para pessoa física

```
Nome: Maria

CPF: 111.111.111-11
```

e para pessoa jurídica

```
Nome: Simples

CNPJ: XX.XXX.XXX/YYYY-ZZ
```

A impressão do nome não deve ser adicionada nas classes filhas, deve ser aproveitada do método imprimir da classe pai (Pessoa)

Faça uma classe que tenha o método main que crie uma pessoa física, uma jurídica e chame o método imprimir dos dois objetos, a impressão deve ser igual a dos exemplos acima.

4) 2pts Crie uma interface **Avaliavel** que terá um método **calcularNota(double nota1, double nota2)** que retornará a média de quem implementá-la.

Crie uma superclasse **Pessoa** que não permitirá ser instanciada. Esta classe terá somente o **nome** (protegido) que deverá ser preenchido em seu construtor.

Crie uma subclasse **Professor** que terá o atributo **materia** e o método **avaliar(Avaliavel avaliavel, double nota1, double nota2),** este método faz chamada para o método

calcularNota(nota1, nota2) da interface e verifica se a média for maior ou igual a 6, imprimindo "Aprovado" se true ou "Reprovado" se false.

Crie uma subclasse **Aluno** que será **Avaliavel** e terá o atributo **matricula.**

Crie uma classe **SalaDeAula** que terá o atributo **professor** e um **mapa de alunos.** Esta classe terá o método **adicionarAluno(Aluno a)** que adicionará um aluno no mapa, na qual a chave será o número da chamada e o valor será o próprio aluno, o método também poderá lançar uma exceção do tipo **LimiteDeAlunosPorSalaException** caso o mapa exceda 30 alunos. A chave do mapa deve ser sequencial (começar em 1 e ir incrementando).

Crie uma classe Principal que terá o método **main.** Instancie a sala de aula, insira um professor e 5 alunos. Depois, faça este professor adicionado, avaliar os 5 alunos, informando quaisquer notas.

Obs: Quando não informado, utilizar o modificador de acesso private para os atributos.

Obs²: Utilizar getters e setters para os atributos somente quando necessário.

5) 2pts Crie uma classe chamada *Casa* com atributos **donoDaCasa** e uma lista de **quartos** (List), também crie a classe *Quarto* com os atributos **cor** e **numero**. Na casa faça o método **adicionaQuarto** que deve receber um quarto como parâmetro e adicionar na lista de quartos, neste mesmo método faça uma verificação na lista para ver se o quarto já existe na casa (verificação por número do quarto), caso existir lançar uma exceção do tipo **checked** (necessita de tratamento), caso não exista, adicionar na lista.

Dentro do bloco main crie e preencha os objetos para que imprima conforme o exemplo abaixo.

```
Quarto já existente na casa


Casa do Fulano tem o(s) quarto(s)

1 - azul, 2 - branco
```

Casa do Clicano tem o(s) quarto(s)

1 - verde, 2 - amarelo

6) Alien Numbers (Retirado da competição Google Code Jam)

O sistema de números decimais é composto por dez dígitos, no qual representamos utilizando "0123456789" (os dígitos em um sistema são escritos do menor para o maior).

Imagine que você descobriu um sistema numeral alienígena composto de um certo número de dígitos, que podem ou não ser os mesmos que os usados em decimal. Por exemplo, se o sistema numeral estrangeiro fosse representado como "oF8", os números de um a dez seriam (F, 8, Fo, FF, F8, 8o, 8F, 88, Foo, FoF). Gostaríamos de poder trabalhar com números em sistemas alienígenas arbitrários (entenda arbitrário como sem regra). Geralmente, queremos ser capazes de converter um número arbitrário que está escrito em um sistema alienígena em um segundo sistema alienígena.

Input

A primeira linha de entrada fornece o número de casos N. Seguidos pelos N casos de teste. Cada caso é uma linha formatada como

numero_alienigina linguagem_origem linguagem_alvo

Cada idioma será representado por uma lista de seus dígitos, ordenada do menor para o maior valor. Nenhum dígito será repetido em qualquer representação, todos os dígitos no número estrangeiro estarão presentes no idioma de origem e o primeiro dígito do número alienígena não será o dígito de menor valor do idioma de origem (em outras palavras, os números estrangeiros não tem zeros à esquerda). Cada dígito será um número 0-9, uma letra maiúscula ou minúscula ou um dos seguintes símbolos !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

Output

Para cada caso de teste, imprima uma linha contendo "Case #x:" seguido do número alienígena traduzido do idioma de origem para o idioma de destino.

Input

```
5
9 0123456789 oF8
```

```
Foo oF8 0123456789
13 0123456789abcdef 01
CODE O!CDE? A?JM!.
prova rapov! icdf!l
```

Output

```
Case #1: Foo
Case #2: 9
Case #3: 10011
Case #4: JAM
Case #5: dif!c
```