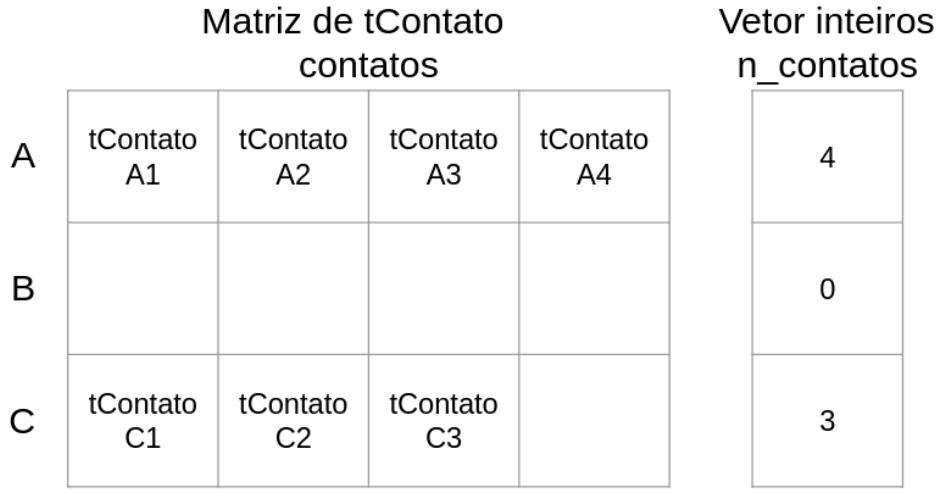
Correção: a questão está dividida em casos visíveis, ocultos, sendo esses simples e completos. O aluno ganhará os pontos conforme a tabela abaixo. Para ganhar os pontos referentes à modularização (implementação dos tipos e suas funções conforme pedido), o aluno deverá acertar pelo menos um dos casos visíveis.

Casos	**Pontos**
Visíveis triviais	0,25
Ocultos triviais	0,25
Visíveis completos	0,25
Ocultos completos	0,25
Modularização	1,5

(**BOCA: P2_Q3**) Problema: Escreva um programa que simule uma agenda telefônica. O programa deve ler da entrada padrão uma lista de contatos, em que cada contato possui os seguintes campos: nome, telefone e e-mail. Para que um contato seja considerado válido, ele deve obrigatoriamente conter o campo nome preenchido e pelo menos um dos outros dois campos (telefone ou e-mail). Casos simples não haverá contatos inválidos. Considere que, sempre que um campo estiver presente, ele estará corretamente formatado; portanto, o programa deve apenas verificar a existência ou ausência desses campos. Após o processamento da lista, o programa deve imprimir na saída padrão todos os contatos da agenda ou apenas uma letra de acordo com o filtro informado (ou seja, imprimir todos os contatos cuja inicial do nome corresponda a uma letra específica informada). Casos simples socilitarão apenas a impressão de toda a agenda.

O código do programa deve ser modularizado, utilizando os tipos tAgenda e tContato. A tAgenda deve armazenar informações gerais sobre uma agenda como lista de contatos e a quantidade de contatos salvo, por exemplo. O aluno deve avaliar os atributos necessários para armazenar a informação pedida. A seguir, é apresentado uma ilustração de uma possível modelagem para a estrutura tAgenda, na qual há uma matriz de tContatos organizada de forma que cada linha armazena os contatos com a mesma letra inicial do nome. Além disso, a estrutura inclui um vetor de inteiros que registra a quantidade de contatos associados a cada letra. O uso dessa modelagem não é obrigatório, você é livre para modelar da forma que considerar mais conveniente.



A utilização e o acesso a tAgenda devem ser feitos através de funções do próprio tipo. Pelo menos as funções abaixo devem ser implementadas:

- **tAgenda tAgenda_adicionaContato(tAgenda agenda, tContato contato);** Recebe uma agenda e um contato como parâmetros. Retornando a agenda atualizada com o contato adicionado caso ele seja válido.
- **void tAgenda_imprime(tAgenda agenda, char filtro);** Recebe uma agenda e um filtro como parâmetros. Imprime os contatos de acordo com o filtro. Quando o filtro assumir o valor asterisco (*), deve ser impresso a agenda completa, se a agenda estiver vazia, não deve ser impresso nada. Caso seja uma letra maiúscula do alfabeto, deve ser impresso todos os contatos que começam com aquela letra. Caso seja pedido para imprimir uma letra maiúscula e a agenda não possua contatos para essa letra, deve ser impresso a seguinte frase: “Agenda vazia para a letra #.”, onde # é a letra do alfabeto em maiúscula que foi pedida para ser impressa.

O tContato deve armazenar informações sobre um contato: nome, telefone e email, por exemplo. A utilização e o acesso a tContato deve ser feita através de funções do próprio tipo. Pelo menos as funções abaixo devem ser implementadas:

- **tContato tContato_le();** Lê as informações de um contato da entrada padrão e retorna um tContato com as informações podendo este ser válido ou não. Assumir que a função será chamada no início da linha que começa o contato e consumirá a quebra de linha;
- **int tContato_validaContato(tContato contato);** Recebe um contato como parâmetro e retorna verdadeiro se é um contato válido, seguindo os requisitos informados anteriormente, caso contrário, retorna falso;

- **char tContato_retornaLetra(tContato contato);** Recebe um contato como parâmetro e o char que corresponde a primeira letra do nome do contato enviado como parâmetro;
- **void tContato_imprime(tContato contato);** Recebe um contato como parâmetro e imprime na saída padrão o nome do contato, telefone e o email, sendo um por linha. O formato deve seguir o padrão:
  Nome
  -Email
  -Telefone
  onde "Nome", "Email" e "Telefone" devem ser substituídos pelas suas respectivas informações salvas do contato. Caso um desses dois campos não exista, ele não deve ser impresso na saída padrão.

Dentro do arquivo compactado, há um arquivo chamado "esqueleto.c" que contém a modelagem sugerida e os cabeçalhos das funções obrigatórias já implementados.

Definição dos formatos de entrada e saída:

- Entrada: A entrada começa com uma linha contendo um número inteiro N ($1 \leq N \leq 100$), que representa a quantidade de contatos a serem adicionados à agenda. As próximas N linhas contêm os contatos a serem adicionados. Cada contato aparecerá no seguinte formato: "[nome,telefone,email]", onde "nome", "telefone" e "email" podem ser preenchidos com uma sequência de caracteres (considerar que a informação não conterá vírgulas ou colchetes para não confundir com o formato) ou com um traço (-), indicando que o campo está vazio. A última linha da entrada contém um filtro de impressão que pode ser letra maiúscula qualquer do alfabeto ou um asterisco (*). A quantidade máxima de contatos por uma letra será 50 (isso permite armazenar os contatos em uma matriz) e o número máximo de caracteres por campos do contato será de 100.

- Saída: A saída do programa deve exibir todos os contatos armazenados na agenda, conforme o filtro de impressão especificado. Se o filtro for uma letra maiúscula do alfabeto, devem ser impressos apenas os contatos cujo nome começa com essa letra. Caso não existam contatos salvos com a letra solicitada, deve ser exibida apenas a mensagem: "Agenda vazia para a letra #.", onde # é a letra solicitada. Se o filtro for um asterisco (*), toda a agenda deve ser impressa seguindo a ordem das letras. Por exemplo, imprimir todos os contatos da letra A na ordem em que foram adicionados, depois todos da letra B também na ordem em que foram adicionados e assim por diante. A impressão de cada contato deve

seguir o formato da função tContato_imprime, com cada campo (nome, telefone e e-mail) exibido em uma linha separada.

Ver exemplos de formato de entrada e saída nos arquivos fornecidos com a questão.

Observação: Recomenda-se utilizar as funções da biblioteca *string.h* (por exemplo strcmp, strlen, strcpy ou outras quando necessário). Por exemplo, a função strcmp compara duas strings retornando 0 quando elas são iguais. Para maiores informações, consultar o manual com *man strcmp*.