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

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

- (**BOCA: P2_Q4**) Problema: Incremente o programa da questão anterior, agora os contatos podem possuir mais um campo chamado endereço. Portanto, um contato passa a ser válido se qualquer um dos três campos (email, telefone ou endereço) existirem. Além disso, também será possível atualizar as informações de um contato. Portanto, agora, antes de adicionar um contato, será necessário verificar se o nome do contato a ser adicionado já está na agenda. Caso o contato já exista, ele deve ser atualizado com as novas informações. Se o contato não existir, então ele deve ser adicionado seguindo os mesmos requisitos da questão anterior.

  O código do programa deve manter-se modularizado. Os tipos da questão anterior devem ser incrementados com os devidos atributos, por exemplo, o tipo *tContato* deve armazenar também o campo endereço. Assim como suas funções para resolver este problema também devem ser alteradas, por exemplo, adaptar as funções de leitura, impressão e validação para o tipo *tContato*, caso seja necessário. O uso e acesso dos tipos devem continuar sendo feitos por meio de suas funções.

  Pelo menos, a seguinte função de *tAgenda* deverá ser implementada adicionalmente:

  - **int tAgenda_buscaIndiceContatoLetra(tAgenda agenda, tContato contato);** Recebe uma agenda e o contato a ser buscado pelo seu nome como parâmetros. Caso o nome esteja na sua respectiva letra da agenda, a

função deve retornar o índice do contato naquela letra da agenda. Caso contrário, ou seja, se o contato não estiver na agenda, a função deve retornar -1;

- Entrada: A entrada começa da mesma forma que a questão anterior com o número inteiro N ($1 \leq N \leq 100$), que representa a quantidade de contatos a serem adicionados à agenda. Da mesma forma, as próximas N linhas contêm os contatos a serem adicionados. As N linhas seguintes contêm os contatos, que podem estar em dois formatos: o primeiro, igual ao do exercício anterior, segue o padrão "[nome,telefone,email]", e o segundo inclui um campo adicional de endereço conforme a seguir, "[nome,telefone,email,endereco]". Os campos podem conter 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. Nos casos triviais, todos os contatos aparecerão no primeiro formato. Já nos casos completos, os dois formatos podem aparecer. A última linha da entrada contém um filtro de impressão que sempre será 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 seguirá o mesmo formato da questão anterior, ou seja, exceto que a impressão deve incluir também o campo endereço quando presente.

  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 e a função strcpy que copia o conteúdo da string de origem para a string de destino. Para maiores informações, consultar o manual com man strcmp ou man strcpy.