Lista de Exercícios 03 – Estruturas e Tipos Abstratos de Dados

3.1 Definir um tipo chamado **Horario** para armazenar um horário composto de: hora, minuto, segundo. Escreva um programa em C para ler dois horários. Escreva o menor horário no formato HH:MM:SS. No caso de igualdade escrever a mensagem "Horários iguais".

3.2 Definir um tipo chamado **Aluno** para armazenar os seguintes dados de um aluno: número de matrícula e data de nascimento (tipo **Data**).
Escrever um programa em C para ler a data de hoje armazenando-a em uma variável do tipo **Data**. A seguir ler uma quantidade indeterminada de dados de alunos (**Aluno**). Para cada aluno lido escrever se ele já completou 18 anos até a data informada. O programa termina ao ser informado um valor negativo para a matrícula. Nesta situação a data de nascimento não deve ser lida.

3.3 Defina o tipo **Conta** para armazenar o número de uma conta e seu respectivo saldo.

Implemente as seguintes operações:

criaConta Saída: Uma conta Entrada: Número da conta Descrição: Atribui os valores iniciais. O saldo deve ser zerado.	**depositaNaConta** Entrada/Saída: Uma conta Entrada: valor do depósito. Descrição: Atualiza o atributo saldo com o valor do depósito.
retiraDaConta Entrada/Saída: Uma conta Entrada: valor da retirada. Descrição: Atualiza o atributo saldo com o valor da retirada.	**obtemSaldo** Entrada: uma conta Retorno: O saldo da conta

Escreva um programa para controlar a conta corrente e a conta poupança do sr. Silva. As contas são integradas de forma que quando não houver saldo suficiente na conta corrente uma transferência automática da conta poupança cobrirá um eventual saldo (o valor da transferência deve ser igual ao valor necessário para cobrir o saldo negativo). O programa deve ler o número e o saldo inicial da conta corrente e da poupança criando duas variáveis do tipo **Conta** (cada uma representa uma conta). A seguir ler uma quantidade indeterminada de duplas de dados representando respectivamente o código da operação (**1**. Depósito conta corrente **2**. Depósito poupança **3**.Retirada conta corrente **4**.Retirada poupança **5**.Fim) e o valor do movimento. O programa termina ao ser informado um código igual a **5** (nesta situação o valor do movimento não deve ser lido).

- Cada conta deve ser armazenada em uma variável do tipo **Conta**.
- A cada operação executada o programa deve exibir o saldo atualizado das duas contas.
- A operação de retirada da conta poupança só deverá ocorrer se houver saldo disponível, caso contrário a mensagem "Saldo insuficiente" deverá ser exibida.
- As operações de depósito e retirada devem ser realizadas com chamadas as funções **depositaNaConta** e **retiraDaConta**.
- A operação de retirada da conta corrente deverá ser executada da seguinte forma:

```
se saldo na conta corrente for insuficiente então
   se saldo na poupança for suficiente para cobrir o valor que falta então
       transferir o valor que falta para a conta corrente e executar a retirada
   senão
       exibir a mensagem "Saldo insuficiente"
   fim_se
senão
   executar a retirada
fim_se
```

3.4 Defina o tipo **Cheque** para armazenar o número, o valor e a situação (0-Em branco, 1-Emitido, 2-Compensado) de um cheque.

Implemente as seguintes operações:
Inclua as definições e protótipos no arquivo **Cheque.h**
A implementação das funções no arquivo **Cheque.c**

criaCheque Saída: um cheque Entrada: Número do cheque Descrição: Atribui os valores iniciais. O valor e a situação do cheque devem ser zerados.	**exibeCheque** Entrada: Um cheque Descrição: Exibe um uma única linha as 3 informações contidas no cheque.
obtemSituacao Entrada: um cheque Retorno: O código da situação do cheque	**obtemValor** Entrada: um cheque Retorno: O valor do cheque
obtemNumero Entrada: um cheque Retorno: O número do cheque	**compensaCheque** Entrada/Saída: um cheque Retorno: Código 0 (sucesso), 1 (erro) Descrição: Altera a situação do cheque para **2** (compensado). O cheque só pode ser compensado se a situação for igual a **1** (emitido)
emiteCheque Entrada/Saída: um cheque Entrada: Valor do cheque Retorno: Código 0 (sucesso), 1 (erro) **Descrição:** Atribui um valor ao cheque e altera a situação para o código 1 (Emitido). O cheque só pode ser emitido se a situação for igual a **0** (em branco)	

Escreva um programa para ler dois números e criar dois cheques armazenando-os em 2 variáveis (usar **criaCheque**).
Ler 2 valores e emitir os 2 cheques (usar **emiteCheque**).
Escrever na tela os dados dos 2 cheques (usar **exibeCheque)**
Compensar o que possui maior valor (usar **obtemValor** para obter o valor de cada cheque e **compensaCheque** para fazer a compensação.
Escreva somente a código da situação dos 2 cheques (usar **obtemSituacao**).
Escreva somente a número dos 2 cheques (usar **obtemNumero**).