Tipo Abstratos de Dados TAD

1) Responda as seguintes questões:

a) O que é um Tipo Abstrato de Dados (TAD) e qual a característica fundamental na sua utilização?

```
R) É uma boa prática de desenvolvimento para funcionalidades com o
intuito de serem compartilháveis, basicamente
a principal característica é a exposição apenas das operações
principais sobre um tipo de dado alvo do TAD,
permitindo assim, que a implementação de uma determinada abstração de
contexto seja deixada de lado, e o foco vá
para o uso das funcionalidades expostas.
```

b) Quais as vantagens de se programar com TADs?

```
R) Além do fato de modularizar a plicação (o que é muito bom do ponto
de vista da organização do projeto), torna
ele mais fácil de manter, ajuda quem usa a manter o foco no que
realmente importa, e como o que fica exposto são
somente as assinaturas de funcionalidades específicas, ajuda a
"esconder" a implementação de cada feature.
```

2) Faça a especificação de um sistema de controle de reservas de um clube que aluga quadras poliesportivas usando TAD utilizando a linguagem C, da seguinte forma:

- No arquivo ".H", declare tudo aquilo que será visível para o programador. *sistema.h: TAD AluguelQuadra*

```c
//Definição do tipo AlguelQuadra
typedef struct aluguel_quadra AluguelQuadra;

// Realiza o aluguel de uma quadra
// (retorna 1, caso dê certo, e 0 caso dê errado)
int reservar(int id_cliente, AluguelQuadra* aq);

// Efetua a devolução de uma quadra alugada
// (retorna 1, caso a quadra tenha sido devolvida, ou 0, caso algo de
errado ocorra)
```

```
int devolver(AluguelQuadra* aq);
```

- No arquivo ".C", defina tudo aquilo que deve ficar oculto do usuário da biblioteca. Para especificação do sistema, não é necessário implementar o corpo das funções.

```
struct aluguel_quadra {
    char nome[50];
    int capacidade;
    double valor;
    int disponivel;
    int idClienteAlugador;
}

// retorna 1: Caso a quadra esteja disponível para locação
// retorna 0: Caso a quadra não esteja disponível para locação
int verificaDisponibilidade(AluguelQuadra* aq);

// verifica a disponibilidade de uma quadra, e:
// retorna 1: caso o aluguel ocorra com sucesso
// retorna 0: caso haja algum problema impedindo a ação
int reservar(int id_cliente, AluguelQuadra* aq);

// Verifica se uma quadra não está disponível para fazer a devolução:
// marca disponível com o valor 0
// define o idClienteAlugador como 0
int devolver(AluguelQuadra* aq);
```

---

3) Desenvolva um TAD para um retângulo. Inclua as funções de inicializações necessárias, as operações para obtenção dos dados armazenados no TAD (base e altura), e a operação que retorne a área do retângulo (A = Base x Altura). Para especificação do sistema, não é necessário implementar o corpo das funções.

triangulo.h

```
// Definindo o tipo Triangulo
typedef struct triangulo Triangulo;

// Cria um triangulo e retorna um ponteiro para o mesmo
Triangulo* criarTriangulo(int ladoA, int ladoB, int ladoC);

// Retorna a medida do lado A do triangulo
int obterLadoA(Triangulo* t);

// Retorna a medida do lado B do triangulo
int obterLadoB(Triangulo* t);
```

```
// Retorna a medida do lado C do triangulo
int obterLadoC(Triangulo* t);
```

triangulo.c

```
include <stdlib.h> // malloc, free, exit
include <stdio.h>  // printf
include "triangulo.h"

// Definindo a struct para o tipo Triangulo
struct triangulo {
    int ladoA;
    int ladoB;
    int ladoC;
};

// Implementação da função que cria um triangulo e retorna um ponteiro
para o mesmo
Triangulo* criarTriangulo(int ladoA, int ladoB, int ladoC){
    ...
}

// Implementação da função que retorna a medida do lado A do triangulo
int obterLadoA(Triangulo* t){
    ...
}

// Implementação da função que retorna a medida do lado B do triangulo
int obterLadoB(Triangulo* t){
    ...
}

// Implementação da função que retorna a medida do lado C do triangulo
int obterLadoC(Triangulo* t){
    ...
}
```

---

4) Crie um TAD para o desenvolvimento de um sistema para representar e gerenciar livros de uma biblioteca. A sua tarefa é implementar um TAD para representar os livros neste sistema. Sabe-se que um livro é representado pelo seguinte tipo estruturado:

```
struct livro {
    char titulo[50];
    char autor[30];
    char genero[10];
```

```
    int ano;
};
```

As funções que devem ser implementadas pelo TAD Livro (na interface do TAD), são as seguintes:

- Função *criaLivro* - que recebe por parâmetro o título, autor, gênero e ano de publicação do livro, cria um livro com esses dados e retorna um ponteiro para o novo Livro.
- Funções de obtenção dos dados armazenados em um TAD Livro (denominadas *obtemGenero*, *obtemAutor*, *obtemTitulo*, e *obtemAno*) que recebem um ponteiro para Livro e retornam o valor em questão.
- Função *verificaNoModernismo* - que recebe um ponteiro para Livro e verifica se esse livro pertence ao segundo período do modernismo Brasileiro (1930 a 1945). Esta função retorna -1 se o ano da obra for anterior a 1930, retorna 0 se for no período 1930 a 1945, e retorna 1 se o ano for posterior a 1945.

livro.h

```
// Definição do tipo Livro
typedef struct livro Livro;

// Cria um livro e retorna um ponteiro para o mesmo
Livro* criaLivro(char* titulo, char* autor, int ano);

// Retorna o gênero de um livro
char* obtemGenero(Livro* l);

// Retorna o nome do autor de um livro
char* obtemAutor(Livro* l);

// Retorna o título de um livro
char* obtemTitulo(Livro* l);

// Retorna o ano de publicação de um livro
int obtemAno(Livro* L);

// Retorna -1: Se o ano da obra for anterior a 1930
// Retorna  1: Se o ano da obra for posterior a 1945
// Retorna  0: Se for no período 1930 a 1945
int verificaNoModernismo();
```

livro.c

```
include <stdlib.h> // malloc, free, exit
include <stdio.h>  // printf
include "livro.h"
// Definição da struct Livro
// ...
```

```
// ...

//
Livro* criaLivro(char* titulo, char* autor, char* genero, int ano){
    Livro* l = (Livro*) mallco(sizeof(Livro));

    // Encerra o programa caso o espaço para o livro não tenha sido
alocado
    if(livro == NULL) {
        printf("Memória insuficiente!\n");
        exit(1);
    }

    // Preenchendo os dados no 'objeto' que terá o endereço retornado
    l->titulo = titulo;
    l->autor = autor;
    l->genero = genero;
    l->ano = ano;

    return l;
}

//
char* obtemGenero(Livro* l){
    return l->genero;
}

//
char* obtemAutor(Livro* l){
    return l->autor;
}

char* obtemTitulo(Livro* l){
    return l->titulo;
}

//
int obtemAno(Livro* l){
    return l->ano;
}

//
int verificaNoModernismo(Livro* l){
    if (l->ano < 1930) {
        return -1;
    } else if(l->ano >= 1945) {
        return 1;
    }

    return 0;
}
```

5) Implemente um programa – completo – que utilize TAD do exercício anterior para cadastrar um livro e após o cadastro verificar se o livro pertence ao período de modernismo Brasileiro.

main.c

```
include <stdio.h>  // printf
include "livro.h"

int main(void){

    // Cadastrando um livro
    Livro* livro = criaLivro("Imagine um titulo para o livro", "Nome
de um autor", "ficção temporária", 1844);

    // Verificando se o livro está no modernismo
    int estaNoModernismo = verificaNoModernismo(livro);

    if(estaNoModernismo == 1){
        printf("A obra é posterior ao modernismo\n");
    } else if(estaNoModernismo == 0) {
        printf("A obra é do modernismo\n");
    } else {
        printf("A obra é anterior ao modernismo\n");
    }

return 0;
}
```