P2 Programação II 2013.1 Departamento de Informática/PUC-Rio

Aluno: _____

Matrícula: _____ Turma: _____

	Valor	Nota
Q1A	2.0	
Q1B	2.0	
Q2	3.0	
Q3	3.0	
Total	10.0	

1. A prova é sem consulta e sem perguntas. A interpretação do enunciado faz parte da prova.
2. A prova deve ser completamente resolvida nas folhas que constam deste caderno, utilizando-se frente e/ou verso;
3. A prova pode ser feita utilizando-se lápis ou caneta (azul ou preta);
4. Dispositivos eletrônicos (celulares, tablets, …) devem ser desligados.
5. Cada questão deve ser resolvida espaço destinado a ela e no verso, quando necessário.

Considere um TAD denominado `ObraLiteraria` que se refere à seguinte estrutura contendo dados e características de uma obra de literatura usada por uma livraria para organizar o seu acervo:

/*Abaixo a definição da struct obraLiteraria que faz parte da implementação do TAD */

```
struct obraLiteraria
{
   char  tipo[10]; /* poesia, teatro, ... */
   char  autor[30];
   char  titulo[50];
   int   ano;
};
```

As funções exportadas por este TAD (na interface do TAD), são as seguintes:

i. A função **obraCria** que recebe os argumentos tipo de obra, autor, título da obra, ano de publicação, cria uma obra literária com esses dados e retorna um ponteiro para essa nova `ObraLiteraria`.

ii. Quatro funções de obtenção dos dados armazenados em um TAD ObraLiteraria (denominadas **obraObtemTipo**, **obraObtemAutor**, **obraObtemTitulo**, e **obraObtemAno**) que recebem um ponteiro para `ObraLiteraria` e retornam o valor em questão (*e.g.* obraObtemTitulo retorna (o ponteiro para) a cadeia com `tipo` de um TAD ObraLiteraria recebido, assim como obraObtemAno retorna o ano de um TAD ObraLiteraria recebido).

iii. Uma sexta função, denominada **obraVerificaNoModernismo2** que recebe um ponteiro para `ObraLiteraria` e verifica se essa obra literária 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.

[Questão 1]

ATENÇÃO: Nos itens da questão 1 solicitados abaixo, o seu papel é o do IMPLEMENTADOR do TAD, ou seja, de quem desenvolveu e disponibilizou o TAD `ObraLiteraria`.

[1a] Escreva o conteúdo do arquivo .h com a interface deste TAD **`ObraLiteraria`**. Coloque como comentário o nome do arquivo e nele apresente a interface completa, isto é, o typedef e todas as funções disponíveis para o usuário.

[1b] Complemente o TAD **`ObraLiteraria`** , escrevendo o módulo (o arquivo .c) que **implementa** as funções deste TAD. Coloque como comentário o nome do arquivo e, para simplificar sua resposta, não precisa apresentar a **obraCria** (que seria necessária, obviamente). Somente apresente a definição da struct e a implementação das funções **obraObtemTipo**, **obraObtemAutor**, **obraObtemTitulo**, **obraObtemAno e a obraVerificaNoModernismo2**. Coloque os comandos #include estritamente necessários.

ATENÇÃO: Nas questões 2 e 3 abaixo, o seu papel é o do USUÁRIO do TAD, ou seja, de quem apenas utiliza o TAD `ObraLiteraria`.

Considere, agora, que o controle do acervo desta livraria é mantido utilizando-se um vetor **v** de ponteiros para `ObraLiteraria` e uma variável inteira n que armazena o número de obras literárias desse acervo.

[Questão 2] Usando o TAD `ObraLiteraria`, escreva a função **ordenaObras** que ordena o vetor **v** em ordem crescente por nome de autor, depois por tipo de obra e finalmente por ano de publicação. Use obrigatoriamente o algoritmo de quick sort. A função recebe um vetor de ponteiros para ObraLiteraria e o número de obras. Para fazer a sua função **ordenaObras** você deve obrigatoriamente usar uma função auxiliar de comparação, para comparar duas obras literárias de acordo com o critério descrito, que deve ser criada em separado por você e estar na prova. Se quiser, você pode usar o `qsort` da biblioteca C na função ordenaObra. Caso você não saiba implementar o quicksort ou usar a qsort, implemente o bolha (valerá metade da questão), mas mesmo assim tem que apresentar e usar a função separada de comparação.

[Questão 3] Usando o TAD `ObraLiteraria`, escreva a função de busca binária **buscaBinObraModerno2** que recebe um vetor de ponteiros para ObraLiteraria, o número de obras, um nome de autor e um tipo de obra, e retorna um ponteiro para a obra literária mais recente do autor, do tipo indicado, no segundo período do modernismo brasileiro. Considere que o vetor está ordenado conforme o critério descrito na questão [2]. Por exemplo, para um vetor com ponteiros para as seguintes estruturas:

```
{"poesia","Cecilia Meireles","Poemas Escritos na India",1962}
{"poesia","Vinicius de Morais","Ariana, a Mulher",1936}
{"poesia","Vinicius de Morais","Novos Poemas",1938}
{"teatro","Vinicius de Morais","Orfeu da Conceição",1954}
```

a função buscaBinObraModerno2 retorna, com os argumentos autor= `"Vinicius de Morais"` e tipo= `"poesia"`, um ponteiro para `{"poesia","Vinicius de Morais","Novos Poemas",1938}`. Nesta questão, você deve tomar cuidado com as repetições (e.g., no exemplo acima, há duas poesias de Vinicius no período 1930-1945, e a função retorna o ponteiro para a primeira). Caso nenhuma obra atendendo à descrição seja encontrada, a função retorna NULL.

Também crie em separado e use obrigatoriamente uma função auxiliar de comparação. Não precisa escrever os comandos #include.

stdio.h:

```
int scanf (char* formato, ...);
int printf (char* formato, ...);
FILE* fopen (char* nome, char* modo);
int fclose (FILE* fp);
int fscanf (FILE* fp, char* formato, ...);
int fprintf (FILE* fp, char* formato, ...);
char*fgets(char* str, int size, FILE* fp);
int sscanf(char* str, char* formato, ...);
```

math.h:

```
double sqrt (double x);
double pow (double x, double exp);
double cos (double radianos);
double sin (double radianos);
```

string.h:

```
int strlen (char* s);
int strcmp (char* s, char *t);
char* strcpy (char* destino, char* fonte);
char* strcat (char* destino, char* fonte);
char* strdup (char* s);
```

stdlib.h:

```
void qsort (void *v, int n, int tam, int (*cmp)(const void*, const void*));
void* malloc (int nbytes);
void free (void* p);
```