UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS - CSHNB

Curso: Sistemas de Informação
Disciplina: Estruturas de dados 2
Docente: Juliana Oliveira de Carvalho
Discente: Marcos André Leal Silva

UNIVERSIDADE FEDERAL DO PIAUÍ - UFPI

Relatório técnico

Picos-PI
04 de janeiro de 2024

Resumo: Este projeto, tem como objetivo demonstrar a prática e implementação de duas Estruturas de Dados: Árvore rubro-negra e árvore 2-3. Essas estruturas foram utilizadas para a resolução de alguns problemas práticos proposto no projeto através da Linguagem de programação C. O mesmo abordará as etapas de cada questão e apresentará suas respectivas implementações, oferecendo uma visão abrangente das aplicações dessas estruturas de dados. Além do mais, serão feitos experimentos para verificar os tempos de execução tanto da árvore rubro-negra quanto da árvore 2-3.

Introdução: As estruturas de dados são um conceito muito importante para a programação e ciência de dados. Tratam-se de maneiras de agregar e organizar dados na memória de um computador ou dispositivo, de forma que façam sentido e proporcionem um bom desempenho ao serem processados. Nesse sentido, estamos considerando dados como sendo blocos de programação que representam algo e têm a função de resolver problemas computacionais. Para isso, eles devem ter a possibilidade de ser simbolizados, armazenados e manipulados.

Uma árvore rubro-negra é um tipo de árvore binária de busca balanceada, uma estrutura de dados usada em ciência da computação, tipicamente para implementar vetores associativos.

Em ciência da computação, uma árvore 2-3 é uma estrutura de dados auto-balanceada, comumente usada para implementar dicionários. Os números significam uma árvore onde cada nó pai (nó interno) tem dois ou três nós filhos. Um nó que só tem 1 elemento de dados, só pode ter dois ou nenhum nó filho e um nó que tem 2 elementos de dados, só pode ter 3 ou nenhum filho.

O objetivo desse projeto é através das estruturas de dados “Árvore rubro-negra” e “Árvore 2-3”, criar duas aplicações capazes de:

- Simular o gerenciamento de uma biblioteca de músicas utilizando alguns parâmetros pré-determinados pelo enunciado. Essa biblioteca será implementada utilizando tanto árvores rubro-negras quanto árvores 2-3. No final, serão feitos experimentos de análise e desempenho, através de algumas buscas, mostrando o caminho percorrido na árvore para encontrar o item e o tempo gasto. A partir da inserção de informações nas árvores, as aplicações têm que serem capazes de realizar buscas e apresentar as informações gravadas nessas estruturas. O usuário poderá, a partir dos conhecimentos de alguns dados cadastrados, realizar a busca dessas informações que serão apresentadas para o mesmo.
- Simular o gerenciamento de memória de um computador. Esse problema será implementado utilizando uma estrutura de dadas chamada “árvore 4-5”, que segue a mesma ideia da árvore 2-3, onde cada nó poderá armazenar até 4 informações e ter até 5 filhos. Esse gerenciamento contará com a inserção de blocos livres e ocupados e a liberação de alguns blocos a partir do cadastro inicial da memória, sempre respeitado a lógica por trás do enunciado e os parâmetros informados na questão. Esse cadastro inicial da memória, requer que informe o estado inicial (ocupado ou livre), e a partir disso, ir informado a quantidade de Mbytes final.

Para a implementação dos projetos, foi utilizado a linguagem de programação C, com seu paradigma de programação imperativo para a criação do código fonte. Através da ferramenta “Visual Studio Code (VS Code)”, esses programas foram criados, testados e executados.

O primeiro programa apresenta duas estruturas de dados em formato de árvores rubro-negras chamadas de “arvartista” e “arvalbum”. A “arvartista” armazena uma variável do tipo Artista com algumas informações (nome, estilo e num_albuns), um endereço para uma árvore rubro-negra contendo informações de cada álbum, duas estruturas para direcionar o lado de cada nó da árvore

(Dir e Esq) e uma variável cor do tipo inteiro para armazenar a cor do nó. Já “arvalbum” armazena uma variável do tipo Album com algumas informações (título, ano, quantidade de músicas), um endereço para uma lista simples de músicas (nome da música, quantidade de minutos e uma estrutura “prox” para apontar o próximo elemento da lista), duas estruturas para direcionar o lado de cada nó da árvore (Dir e Esq) e uma variável cor do tipo inteiro para armazenar a cor do nó.

E para realizar os testes de tempo de busca da árvore rubro-negra, foi feito um arquivo a parte para demonstrar os resultados do tempo de busca.

O segundo programa segue a mesma lógica do primeiro, mas implementado com árvore 2-3. Contudo, é adicionado um novo ponteiro “centro” nas estruturas de “arvartista” e “arvalbum” para apontar para um filho localizado ao centro desse bloco.

O terceiro programa apresenta uma estrutura de dados em formato de árvore 4-5 chamada “Arv45”. “Arv45” armazena 4 variáveis do tipo Bloco com algumas informações(status, Bloco inicial e Bloco final), uma variável do tipo inteiro para armazenar a quantidade de informações e cinco estruturas para direcionar o lado de cada nó da árvore (Esq, cen_esq, cen, cen_dir, Dir).

Seções específicas:

Informações técnicas: Para o desenvolvimento e testes deste projeto foi utilizado um notebook com um processador 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz, doze gigas de memória RAM ddr4, sistema operacional com a arquitetura de 64 bits, Windows 11 Home.

Os códigos foram feitos na IDE e editor de texto Visual Studio Code e compilados pelo compilador gcc (MinGW.org GCC Build-2) 9.2.0.

Questão 1:

O programa implementa uma estrutura de árvore de busca binária (BST) e uma árvore rubro-negra (LLRB) para armazenar informações sobre artistas, álbuns e músicas. As estruturas de dados definidas são Artista, Album, e Musica, e as árvores são implementadas como ArvArtista, ArvAlbum, ArvLLRB, e ArvLLRBalbum.

1. Leitura de Dados:

- Funções ler_artista, ler_album e ler_musica são usadas para obter informações sobre artistas, álbuns e músicas, respectivamente.

2. Inserção e Remoção de Artistas, Álbuns e Músicas:

- As funções inserir_artista, insereArvLLRB_artista, excluir_artista são responsáveis por inserir e excluir artistas na árvore de busca binária e na árvore rubro-negra.
- As funções inserir_album, insereArvLLRB_album, excluir_album são responsáveis por inserir e excluir álbuns na árvore de busca binária e na árvore rubro-negra.
- A função inserirMusica é utilizada para inserir músicas em uma lista associada a um álbum.

3. Busca de Artistas, Álbuns e Músicas:

- As funções buscaArvSArtista, buscaArvSAlbum, e busca_musica realizam buscas nas respectivas árvores para encontrar artistas, álbuns e músicas.

4. Impressão de Dados:

- As funções imprimir_um, imprimir_todos_artistas, imprimir_um_album e imprimir_todos_albums são usadas para imprimir informações sobre artistas e álbuns.

5. Balanceamento da Árvore Rubro-Negra:

- Funções como rotacionaEsquerdaArtista, rotacionaDireitaArtista, move2EsqREDArtista, move2DirREDArtista, balancear_artista são responsáveis por garantir que a árvore rubro-negra mantenha suas propriedades após inserções e exclusões.

6. Liberação de Memória:

- Funções como liberarMusicas, liberarAlbums, liberarArtistas, liberarTodaMemoria são usadas para liberar a memória alocada para as listas de músicas e para as árvores de artistas e álbuns.

Questão 2:

O programa implementa uma árvore B de ordem 2-3 (Arv23) para gerenciar informações sobre biblioteca de músicas.

Estrutura de dados:

1. Artista (struct artista):
- Armazena informações sobre um artista, como nome, estilo e número de álbuns.

2. Album (struct album):
- Contém detalhes sobre um álbum, como título, ano e quantidade de músicas.

3. Musica (struct musica):
- Representa uma música, com nome e duração em minutos, e possui um ponteiro para a próxima música na lista.

4. ArvArtista (struct arvartista):
- Árvore B que armazena informações sobre artistas.
- Contém ponteiros para informações de artistas, quantidade de chaves (Nkeys), ponteiro para árvore de álbuns e ponteiros para os nós filhos (esquerdo, central e direito).

5. ArvAlbum (struct arvalbum):
- Árvore B que armazena informações sobre álbuns.
- Contém ponteiros para informações de álbuns, lista de músicas, quantidade de chaves (Nkeys) e ponteiros para os nós filhos.

Funções Principais:

Para Árvore de Artistas (ArvArtista):

criaArvArtista():
Cria uma nova árvore de artistas vazia.

insereArvArtista():
Insere um novo artista na árvore.

imprimirArvArtista():
Realiza uma impressão (possivelmente em ordem) da árvore de artistas.

removerArtista():
Remove um artista da árvore.

buscaArvArtista():

Busca por um artista na árvore.

Para Árvore de Álbuns (ArvAlbum):

criaArvAlbum():
Cria uma nova árvore de álbuns vazia.

insereArvAlbum():
Insere um novo álbum na árvore.

imprimirArvAlbum():
Realiza uma impressão (possivelmente em ordem) da árvore de álbuns.

removerAlbum():
Remove um álbum da árvore.

buscaArvAlbum():
Busca por um álbum na árvore.

Para Músicas (Musica):

criarLista():
Cria uma nova lista de músicas vazia.

inserirMusica():
Insere uma nova música na lista de músicas associada a um álbum.

busca_musica():
Busca por uma música em uma lista de músicas.

excluir_musica():
Exclui uma música de uma lista de músicas.

imprimir_todas_musicas():
Imprime todas as músicas de uma lista.

Questão 3:

O programa implementa uma árvore B de ordem 4-5 (Arv45) para gerenciar informações sobre blocos de memória.

1. Estruturas de Dados:

- Bloco: Estrutura que armazena informações sobre um bloco de memória, incluindo o início (Bloco_I), fim (Bloco_F), e status (status).
- Arv45: Estrutura que representa um nó da árvore 4-5. Contém quatro informações (info1, info2, info3, info4) e cinco ponteiros para subárvores (Esq, Cen_Esq, Cen, Cen_Dir, Dir). Ninfos é uma variável que indica quantas informações estão presentes no nó

.

2. Função Cadastra:

- Solicita ao usuário a quantidade de blocos lógicos da memória em Mbytes (Tam_bloco) e o estado do primeiro bloco.
- Utiliza um loop para cadastrar informações sobre os blocos e inseri-los na árvore 4-5.

3. Função insere:

- Realiza a inserção de um novo bloco na árvore 4-5, mantendo a estrutura da árvore balanceada.
- Divide os nós quando necessário, criando novos nós e promovendo informações.

4. Função criaNo:
- Aloca memória para criar um novo nó da árvore 4-5 e inicializa suas informações e ponteiros.

5. Função ehFolha:
- Verifica se um nó é uma folha da árvore.

6. Função adicionaChave:
- Adiciona uma nova informação a um nó da árvore, considerando as diferentes situações de inserção.

7. Função quebraNo:
- Divide um nó da árvore 4-5 quando necessário, criando um novo nó e promovendo uma informação.

8. Função exibirInfo:
- Exibe as informações de um bloco na tela.

9. Função imprimir:
- Percorre a árvore em ordem e exibe as informações de todos os blocos.

10. Função AlocaEspacos:
- Utiliza a função ProcuraEspaco para encontrar um bloco de memória disponível para alocar a quantidade desejada de blocos.

11. Função busca45:
- Realiza uma busca na árvore para encontrar um bloco com início igual ao valor informado.

12. Função remover:
- Remove um bloco da árvore 4-5, mantendo a estrutura balanceada.

Chama a função balancear quando necessário.

13. Função balancear:
- Realiza o balanceamento da árvore após a remoção de um bloco.

14. Função removeInfo2Folha:
- Remove a segunda informação de um nó folha.

15. Função removeInfo2ComDirFolha:
- Remove a segunda informação de um nó que possui uma subárvore à direita.

16. Função obterMenorNo:
- Obtém o nó com a menor informação em uma subárvore.

17. Função numeroInfosArv:
- Calcula o número total de informações presentes na árvore.

18. Função obterMaiorNo:
- Obtém o nó com a maior informação em uma subárvore.

19. Função removeInfo2NaoFolha:
- Remove a segunda informação de um nó não folha.

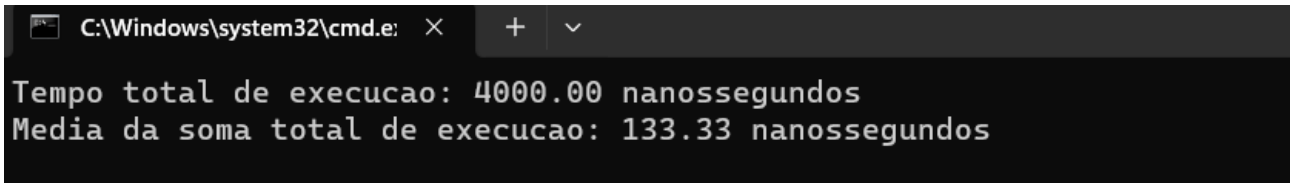
20. Função removeInfo1Folha:
- Remove a primeira informação de um nó folha.

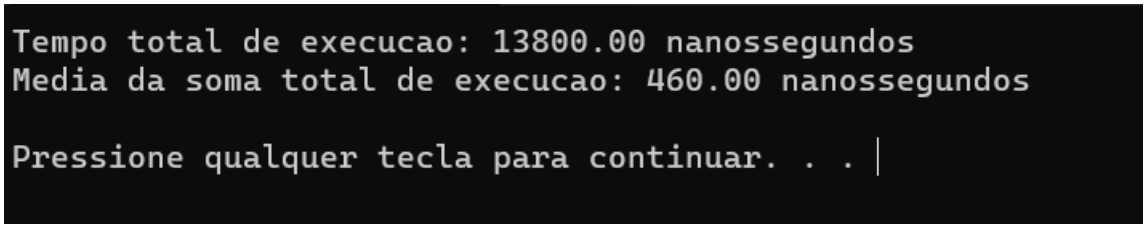
21. Função removeInfo1NaoFolha:
- Remove a primeira informação de um nó não folha.

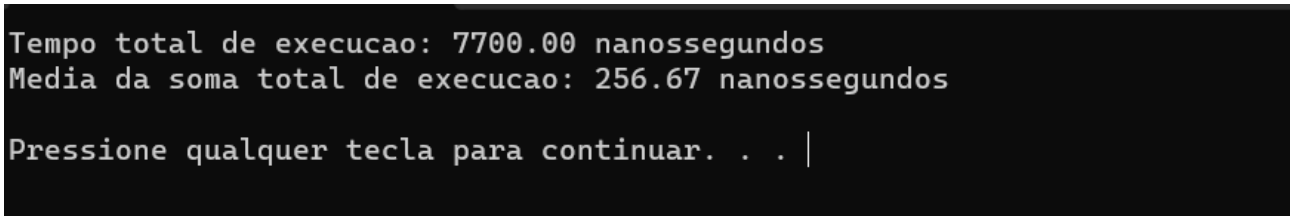
22. Função removeInfo2ComDirFolha:
- Remove a segunda informação de um nó que possui uma subárvore à direita.

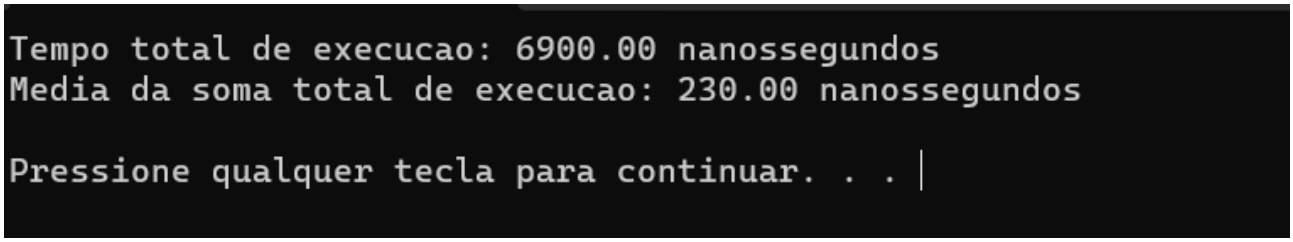
Resultados da execução do programa:

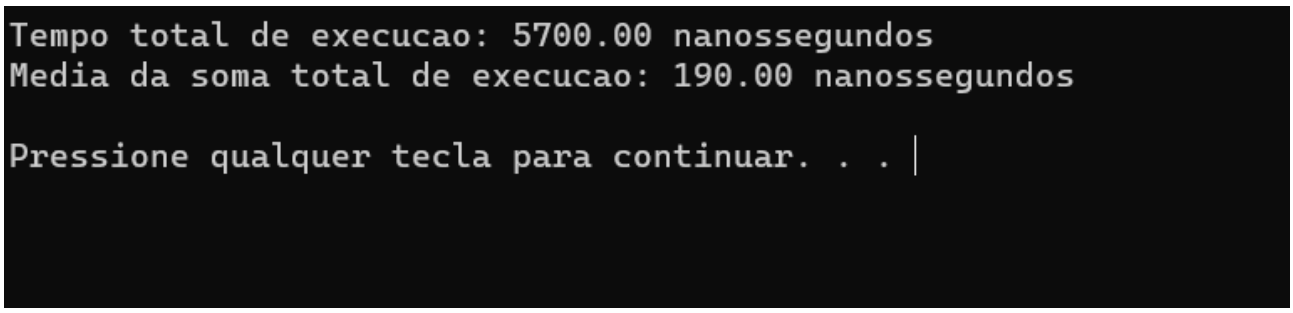
Questão 1.1 (rubro-negra):





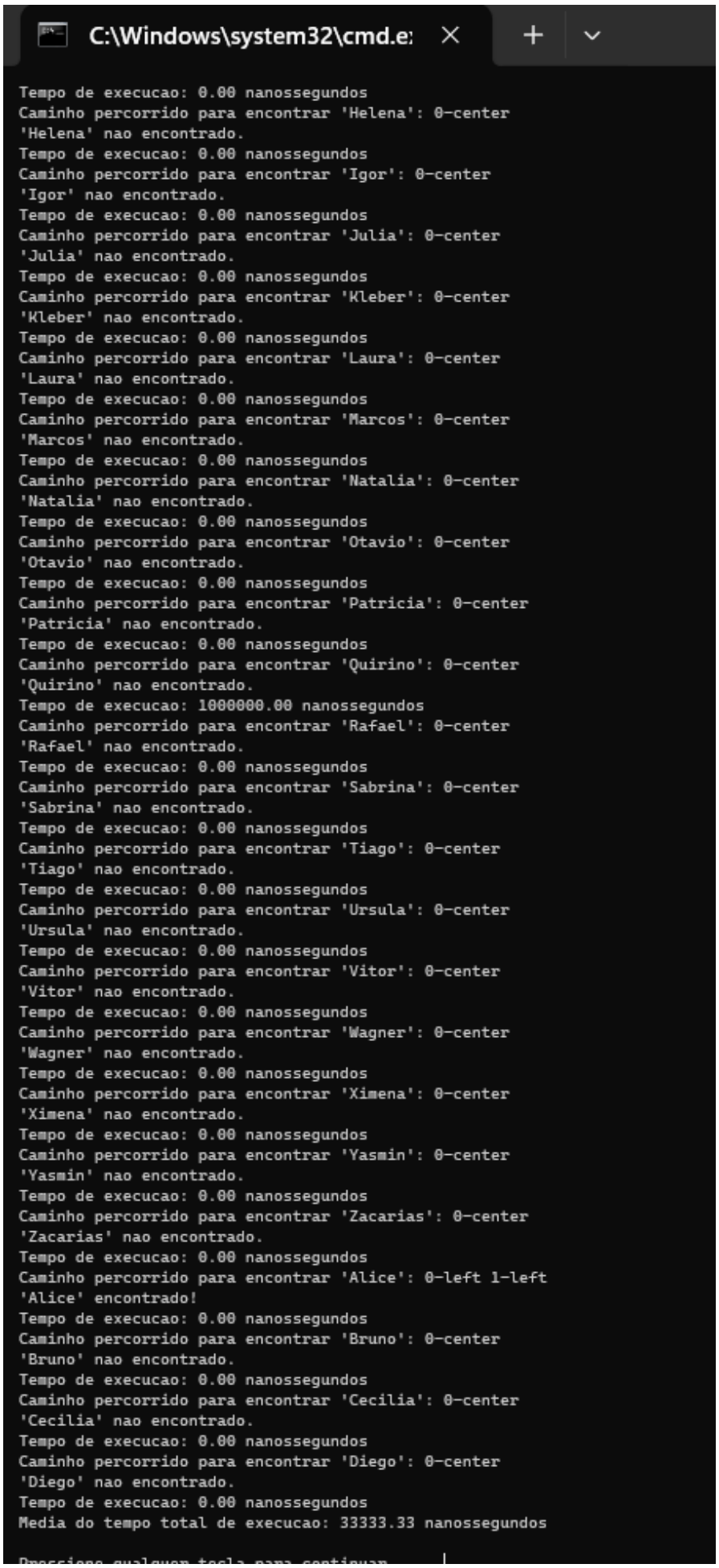






Questão 2.1 (2-3):


C:\Windows\system32\cmd.e:
'Gabriel' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Helena': 0-center
'Helena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Igor': 0-center
'Igor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Julia': 0-center
'Julia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Kleber': 0-center
'Kleber' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Laura': 0-center
'Laura' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Marcos': 0-center
'Marcos' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Natalia': 0-center
'Natalia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Otavio': 0-center
'Otavio' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Patricia': 0-center
'Patricia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Quirino': 0-center
'Quirino' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Rafael': 0-center
'Rafael' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Sabrina': 0-center
'Sabrina' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Tiago': 0-center
'Tiago' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ursula': 0-center
'Ursula' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Vitor': 0-center
'Vitor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Wagner': 0-center
'Wagner' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ximena': 0-center
'Ximena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Yasmin': 0-center
'Yasmin' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Zacarias': 0-center
'Zacarias' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Alice': 0-left 1-left
'Alice' encontrado!
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Bruno': 0-center
'Bruno' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Cecilia': 0-center
'Cecilia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Diego': 0-center
'Diego' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Media do tempo total de execucao: 66666.67 nanossegundos


C:\Windows\system32\cmd.e:
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Helena': 0-center
'Helena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Igor': 0-center
'Igor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Julia': 0-center
'Julia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Kleber': 0-center
'Kleber' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Laura': 0-center
'Laura' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Marcos': 0-center
'Marcos' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Natalia': 0-center
'Natalia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Otavio': 0-center
'Otavio' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Patricia': 0-center
'Patricia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Quirino': 0-center
'Quirino' nao encontrado.
Tempo de execucao: 1000000.00 nanossegundos
Caminho percorrido para encontrar 'Rafael': 0-center
'Rafael' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Sabrina': 0-center
'Sabrina' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Tiago': 0-center
'Tiago' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ursula': 0-center
'Ursula' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Vitor': 0-center
'Vitor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Wagner': 0-center
'Wagner' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ximena': 0-center
'Ximena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Yasmin': 0-center
'Yasmin' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Zacarias': 0-center
'Zacarias' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Alice': 0-left 1-left
'Alice' encontrado!
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Bruno': 0-center
'Bruno' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Cecilia': 0-center
'Cecilia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Diego': 0-center
'Diego' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Media do tempo total de execucao: 33333.33 nanossegundos

```
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Kleber': 0-center
'Kleber' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Laura': 0-center
'Laura' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Marcos': 0-center
'Marcos' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Natalia': 0-center
'Natalia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Otavio': 0-center
'Otavio' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Patricia': 0-center
'Patricia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Quirino': 0-center
'Quirino' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Rafael': 0-center
'Rafael' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Sabrina': 0-center
'Sabrina' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Tiago': 0-center
'Tiago' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ursula': 0-center
'Ursula' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Vitor': 0-center
'Vitor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Wagner': 0-center
'Wagner' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ximena': 0-center
'Ximena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Yasmin': 0-center
'Yasmin' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Zacarias': 0-center
'Zacarias' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Alice': 0-left 1-left
'Alice' encontrado!
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Bruno': 0-center
'Bruno' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Cecilia': 0-center
'Cecilia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Diego': 0-center
'Diego' nao encontrado.
Tempo de execucao: 2000000.00 nanossegundos
Media do tempo total de execucao: 166666.67 nanossegundos
```

```
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Natalia': 0-center
'Natalia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Otavio': 0-center
'Otavio' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Patricia': 0-center
'Patricia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Quirino': 0-center
'Quirino' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Rafael': 0-center
'Rafael' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Sabrina': 0-center
'Sabrina' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Tiago': 0-center
'Tiago' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ursula': 0-center
'Ursula' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Vitor': 0-center
'Vitor' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Wagner': 0-center
'Wagner' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Ximena': 0-center
'Ximena' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Yasmin': 0-center
'Yasmin' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Zacarias': 0-center
'Zacarias' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Alice': 0-left 1-left
'Alice' encontrado!
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Bruno': 0-center
'Bruno' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Cecilia': 0-center
'Cecilia' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Caminho percorrido para encontrar 'Diego': 0-center
'Diego' nao encontrado.
Tempo de execucao: 0.00 nanossegundos
Media do tempo total de execucao: 0.00 nanossegundos

Pressione qualquer tecla para continuar. . .
```

Em geral, para conjuntos de dados pequenos e operações de busca predominantes, a árvore Rubro-Negra pode ser mais eficiente devido à sua estrutura mais simples. Para conjuntos de dados grandes e operações de leitura e gravação frequentes, a árvore B-2-3 pode ter vantagens devido ao seu melhor aproveitamento de bloco e à possibilidade de armazenar mais informações por nó.

O desempenho específico pode variar dependendo da implementação exata, do tamanho do conjunto de dados, dos padrões de acesso e de outros fatores específicos do seu caso de uso. A melhor abordagem é realizar testes de desempenho com dados reais ou simulados para determinar qual árvore é mais eficiente para o seu cenário específico.

Foram buscados um total de 30 elementos em cada árvore, pode-se notar que em alguns casos de busca da árvore 2-3 essa media de tempo foi tão pequeno que chegou a 0.00000000...

nanosegundos. Um dos fatores que podem ter levado a isso é a capacidade da máquina que o programa foi executado.

Conclusão: Os programas forneceram uma estrutura básica para gerenciar bibliotecas de músicas e a memória de um computador por meio da ideia de divisão de blocos lógicos. Além do mais, os experimentos de verificação de tempo de busca, se mostraram ser eficientes perante a máquina utilizada para testar os mesmos. Com tudo que já foi dito, podemos concluir que os programas se mostram eficiente nos resultados de saída exibidos. Todas demais informações já foram citadas, explanadas e detalhadas nos tópicos acima. Sendo assim, os programas conseguiram responder as problemáticas dos projetos e se mostram sem erros ou insuficiências. Com estes experimentos, foi possível desenvolver e manipular árvores rubro-negras, árvores 2-3 e árvores 4-5 através da Linguagem C, apesar de ser um projeto simples, a boa interpretação do problema é crucial para uma ótima aplicação.

Apêndice:

Questão 1:

```
/*  Faça um programa em C de uma Biblioteca de Música. As informações são
organizadas por
Artista (cantor(a), Dupla, Banda, Grupo, …). E para cada artista deve ser ter o
nome do artista, o estilo
musical, o número de álbuns, e os Álbuns (endereço da árvore vermelho e preta).
Cada álbum deve ter, o
título, o ano de lançamento, a quantidade de músicas e as Músicas (endereço lista
ordenada). Para cada
música deve se ter o título, e a quantidade de minutos. Quando o usuário abre o
programa o mesmo deve
automaticamente criar uma árvore vermelha-preta, contendo informações sobre os
Artistas (organizado pelo
nome), e inclui como chave o nome do artista.

    O programa deve permitir a inserção de Artistas, Álbuns e Músicas. Lembrando
que uma música, deve estar
em um álbum e um álbum deve estar em um Artista, ou seja, uma música só pode ser
inserida em um álbum
já cadastrado e um álbum só pode ser inserido para um Artista já cadastrado. O
programa deve permitir
todos os tipos de busca: por artista, por álbum por música. E também permitir a
remoção de uma
determinada música, e de um álbum (neste caso lembrar ao usuário que todas as
músicas daquele álbum
serão removidas e o mesmo referente aos artistas, para a remoção de um artista,
lembrar ao usuário que
todos os álbuns e consequentemente todas as músicas daquele artista serão
removidas. Faça um
experimento que busque por 30 itens(artistas), mostre o caminho percorrido na
árvore para encontrar o item
e o tempo gasto. Depois faça uma análise dos resultados obtidos.
*/
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define RED 1
#define BLACK 0

/*Prototipos*/
typedef struct arvartista ArvArtista;
typedef struct arvllrb ArvLLRB;
typedef struct artista Artista;
typedef struct arvalbum ArvAlbum;
typedef struct album Album;
typedef struct musica Musica;
/*-------------------------------------------------------------------------------------------*/
ArvArtista* criaArvArtista();
int cor_artista(ArvArtista *raiz);
void trocaCor_artista(ArvArtista *raiz);
void rotacionaEsquerdaArtista(ArvArtista **A);
void rotacionaDireitaArtista(ArvArtista **A);
void move2EsqREDArtista(ArvArtista **raiz);
void move2DirREDArtista(ArvArtista **raiz);
void balancear_artista(ArvArtista **raiz);
ArvArtista *removerMenor(ArvArtista *raiz);
ArvArtista *procuraMenor(ArvArtista *atual);
ArvArtista *inserir_artista(ArvArtista *raiz, Artista a, int *resp);
int insereArvLLRB_artista(ArvArtista **raiz, Artista a);
ArvArtista *excluir_artista(ArvArtista *raiz, char *nome);
ArvArtista* buscaArvSArtista(ArvArtista *raiz, char *nome, int *achou);

/*-------------------------------------------------------------------------------------------*/
ArvAlbum* criaArvAlbum();
int cor_album(ArvAlbum *raiz);
void trocaCor_album(ArvAlbum *raiz);
void rotacionaEsquerdaAlbum(ArvAlbum **A);
void rotacionaDireitaAlbum(ArvAlbum **A);
void move2EsqREDAlbum(ArvAlbum **raiz);
void move2DirREDAlbum(ArvAlbum **raiz);
void balancear_album(ArvAlbum **raiz);
ArvAlbum *removerMenorAlbum(ArvAlbum *raiz);
ArvAlbum *procuraMenorAlbum(ArvAlbum *atual);
ArvAlbum *inserir_album(ArvAlbum *raiz, Album a, int *resp);
int insereArvLLRB_album(ArvAlbum **raiz, Album a);
ArvAlbum *excluir_album(ArvAlbum *raiz, char *titulo);
void imprimir_um_album(Album a);
void imprimir_todos_album(ArvAlbum *raiz);
```

```c
ArvAlbum* buscaArvSAlbum(ArvAlbum *raiz, char *titulo, int *achou);
/*-------------------------------------------------------------------------------------
-------------*/
Musica *criarLista();
void inserirMusica(ArvAlbum *lista);
Musica *busca_musica(ArvAlbum *lista, char *nome);
void excluir_musica(ArvAlbum *lista, char *nome, int *excluiu);
void imprimir_todas_musicas(ArvAlbum *lista);
void imprimir_uma_musica(Musica* m);
/*-------------------------------------------------------------------------------------
-------------*/
void liberarTodaMemoria(ArvArtista *raiz);
void liberarArtistas(ArvArtista *raiz);
void liberarAlbums(ArvAlbum *raiz);
void liberarMusicas(Musica *lista);
/*Structs*/
typedef struct artista
{
    char nome[50];
    char estilo[50];
    int num_albuns;
}Artista;

typedef struct album
{
    char titulo[50];
    int ano;
    int quantidade_musica;
}Album;

typedef struct musica
{
    char nome_musica[50];
    int minutos;
    struct musica *prox;
}Musica;


/*Arvores*/

typedef struct arvartista
{
    Artista artista;
    ArvAlbum *arvAlbum;
    struct arvartista *esq, *dir;
    int cor;
}ArvArtista;

typedef struct arvllrb
{
    Artista artista;
```

```
    ArvAlbum *arvAlbum;
    struct arvartista *esq, *dir;
    int cor;
}ArvLLRB;

typedef struct arvalbum
{
    Album album;
    Musica *musica; //lista
    struct arvalbum *esq, *dir;
    int cor;
}ArvAlbum;

typedef struct arvllrbalbum
{
    Album album;
    Musica *musica; //lista
    struct arvalbum *esq, *dir;
    int cor;
}ArvLLRBalbum;

/*Leituras*/

Artista ler_artista()
{
    Artista art;

    fflush(stdin);
    printf("Informe o nome do artista: ");
    scanf("%[^\n]s", art.nome);
    fflush(stdin);
    printf("Informe o estilo musical: ");
    scanf("%[^\n]s", art.estilo);
    fflush(stdin);
    art.num_albuns = 0;
    fflush(stdin);
    return art;
}

Album ler_album()
{
    Album album;
    fflush(stdin);
    printf("Informe o nome do album: ");
    scanf("%[^\n]s", album.titulo);
    fflush(stdin);
    printf("Informe o ano do album: ");
    scanf("%d", &album.ano);
    fflush(stdin);
    album.quantidade_musica = 0;
```

```
    return album;
}

Musica ler_musica()
{
    Musica musica;
    fflush(stdin);
    printf("Informe o nome da musica: ");
    scanf("%[^\n]s", musica.nome_musica);
    fflush(stdin);
    printf("Informe quantidade de minutos: ");
    scanf("%d", &musica.minutos);
    return musica;
}

/*Implementação das Arvores*/

/*Artistas*/

ArvArtista* criaArvArtista()
{
    return NULL;
}

int cor_artista(ArvArtista *raiz)
{
    if (raiz == NULL)
        return BLACK;
    else
        return raiz->cor;
}

void trocaCor_artista(ArvArtista *raiz)
{
    raiz->cor = !raiz->cor;
    if (raiz->esq != NULL)
        raiz->esq->cor = !raiz->esq->cor;
    if(raiz->dir != NULL)
        raiz->dir->cor = !raiz->dir->cor;
}

void rotacionaEsquerdaArtista(ArvArtista **A)
{
    ArvArtista *B = (*A)->dir;
    (*A)->dir = B->esq;
    B->esq = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}
```

```c
void rotacionaDireitaArtista(ArvArtista **A)
{
    ArvArtista *B = (*A)->esq;
    (*A)->esq = B->dir;
    B->dir = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}

void move2EsqREDArtista(ArvArtista **raiz)
{
    trocaCor_artista(*raiz);
    if (cor_artista((*raiz)->dir->esq) == RED)
    {
        rotacionaDireitaArtista(&(*raiz)->dir);
        rotacionaEsquerdaArtista(&(*raiz));
        trocaCor_artista(*raiz);
    }
}

void move2DirREDArtista(ArvArtista **raiz)
{
    trocaCor_artista(*raiz);
    if (cor_artista((*raiz)->esq->esq) == RED)
    {
        rotacionaDireitaArtista(&(*raiz));
        trocaCor_artista(*raiz);
    }
}

void balancear_artista(ArvArtista **raiz)
{
    if (cor_artista((*raiz)->dir) == RED)
        rotacionaEsquerdaArtista(raiz);
    if((*raiz)->esq != NULL && cor_artista((*raiz)->esq) == RED &&
cor_artista((*raiz)->esq->esq) == RED)
        rotacionaDireitaArtista(raiz);
    if (cor_artista((*raiz)->esq) == RED && cor_artista((*raiz)->dir) == RED)
        trocaCor_artista(*raiz);

}

ArvArtista *removerMenor(ArvArtista *raiz)
{
    if ((raiz->esq) == NULL)
    {
        free(raiz);
        return NULL;
    }
    if (cor_artista(raiz->esq) == BLACK && cor_artista(raiz->esq->esq) == BLACK)
```

```
        move2EsqREDArtista(&raiz);

    removerMenor(raiz->esq);

    balancear_artista(&raiz);

    return raiz;
}

ArvArtista *procuraMenor(ArvArtista *atual)
{
    ArvArtista *no1 = atual;
    ArvArtista *no2 = atual->esq;
    while (no2 != NULL)
    {
        no1 = no2;
        no2 = no2->esq;
    }
    return no1;
}

ArvArtista *inserir_artista(ArvArtista *raiz, Artista a, int *resp)
{
    if (raiz == NULL)
    {
        ArvArtista *novo;
        novo = (ArvArtista*)malloc(sizeof(ArvArtista));
        if(novo == NULL)
        {
            *resp = 0;
            return NULL;
        }
        novo->artista = a;
        novo->cor = RED;
        novo->dir = NULL;
        novo->esq = NULL;
        novo->arvAlbum = criaArvAlbum();
        *resp = 1;
        return novo;
    }
    if(strcmp(a.nome, raiz->artista.nome) == 0)
        *resp = 0;
    else
    {
        if (strcmp(a.nome, raiz->artista.nome) < 0)
            raiz->esq = inserir_artista(raiz->esq, a, resp);
        else
            raiz->dir = inserir_artista(raiz->dir, a, resp);
    }
    if (cor_artista(raiz->dir) == RED && cor_artista(raiz->esq) == BLACK)
        rotacionaEsquerdaArtista(&raiz);
```

```
    if (cor_artista(raiz->esq) == RED && cor_artista(raiz->esq->esq) == RED)
        rotacionaDireitaArtista(&raiz);
    if(cor_artista(raiz->dir) == RED && cor_artista(raiz->esq) == RED)
        trocaCor_artista(raiz);

    return raiz;

}

int insereArvLLRB_artista(ArvArtista **raiz, Artista a)
{
    int resp;
    *raiz = inserir_artista(*raiz, a, &resp);
    if (*raiz != NULL)
        (*raiz)->cor = BLACK;

    return resp;
}

ArvArtista *excluir_artista(ArvArtista *raiz, char *nome)
{
    if (raiz == NULL) {
        return NULL;
    }

    if (strcmp(nome, raiz->artista.nome) < 0)
        raiz->esq = excluir_artista(raiz->esq, nome);
    else if (strcmp(nome, raiz->artista.nome) > 0)
        raiz->dir = excluir_artista(raiz->dir, nome);
    else
    {

        if (raiz->esq == NULL && raiz->dir == NULL)
        { //sem filho

            free(raiz);
            return NULL;
        }
        else if (raiz->esq == NULL || raiz->dir == NULL)
        { //um filho

            ArvArtista *temp = (raiz->esq != NULL) ? raiz->esq : raiz->dir;
            free(raiz);
            return temp;
        }
        else
        { //dois filhos
            ArvArtista *temp = procuraMenor(raiz->dir);
            raiz->artista = temp->artista;
            raiz->dir = excluir_artista(raiz->dir, temp->artista.nome);
        }
```

```
    }

    balancear_artista(&raiz);

    return raiz;
}

ArvArtista* buscaArvSArtista(ArvArtista *raiz, char *nome, int *achou)
{
    if (raiz != NULL)
    {
        if (strcmp(nome, raiz->artista.nome) < 0)
            return buscaArvSArtista(raiz->esq, nome, achou);
        else if (strcmp(nome, raiz->artista.nome) > 0)
            return buscaArvSArtista(raiz->dir, nome, achou);
        else
        {
            *achou = 1;
            return raiz;
        }
    }

    *achou = 0;
    return NULL;
}

void imprimir_um(Artista a)
{
    printf("Nome: %s\nEstilo: %s\nNumero de albuns: %d\n\n", a.nome, a.estilo,
a.num_albuns);
}

void imprimir_todos_artistas(ArvArtista *raiz)
{
    if (raiz != NULL)
    {
        imprimir_todos_artistas(raiz->esq);
        imprimir_um(raiz->artista);
        imprimir_todos_artistas(raiz->dir);
    }

}

/*Albuns*/

ArvAlbum* criaArvAlbum()
{
    return NULL;
}
```

```c
int cor_album(ArvAlbum *raiz)
{
    if (raiz == NULL)
        return BLACK;
    else
        return raiz->cor;
}

void trocaCor_album(ArvAlbum *raiz)
{
    raiz->cor = !raiz->cor;
    if (raiz->esq != NULL)
        raiz->esq->cor = !raiz->esq->cor;
    if(raiz->dir != NULL)
        raiz->dir->cor = !raiz->dir->cor;
}

void rotacionaEsquerdaAlbum(ArvAlbum **A)
{
    ArvAlbum *B = (*A)->dir;
    (*A)->dir = B->esq;
    B->esq = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}

void rotacionaDireitaAlbum(ArvAlbum **A)
{
    ArvAlbum *B = (*A)->esq;
    (*A)->esq = B->dir;
    B->dir = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}

void move2EsqREDAlbum(ArvAlbum **raiz)
{
    trocaCor_album(*raiz);
    if (cor_album((*raiz)->dir->esq) == RED)
    {
        rotacionaDireitaAlbum(&(*raiz)->dir);
        rotacionaEsquerdaAlbum(&(*raiz));
        trocaCor_album(*raiz);
    }
}

void move2DirREDAlbum(ArvAlbum **raiz)
{
    trocaCor_album(*raiz);
```

```
    if (cor_album((*raiz)->esq->esq) == RED)
    {
        rotacionaDireitaAlbum(&(*raiz));
        trocaCor_album(*raiz);
    }
}

void balancear_album(ArvAlbum **raiz)
{
    if (cor_album((*raiz)->dir) == RED)
        rotacionaEsquerdaAlbum(raiz);
    if((*raiz)->esq != NULL && cor_album((*raiz)->esq) == RED && cor_album((*raiz)-
>esq->esq) == RED)
        rotacionaDireitaAlbum(raiz);
    if (cor_album((*raiz)->esq) == RED && cor_album((*raiz)->dir) == RED)
        trocaCor_album(*raiz);

}

ArvAlbum *removerMenorAlbum(ArvAlbum *raiz)
{
    if ((raiz->esq) == NULL)
    {
        free(raiz);
        return NULL;
    }
    if (cor_album(raiz->esq) == BLACK && cor_album(raiz->esq->esq) == BLACK)
        move2EsqREDAlbum(&raiz);

    removerMenorAlbum(raiz->esq);

    balancear_album(&raiz);

    return raiz;
}

ArvAlbum *procuraMenorAlbum(ArvAlbum *atual)
{
    ArvAlbum *no1 = atual;
    ArvAlbum *no2 = atual->esq;
    while (no2 != NULL)
    {
        no1 = no2;
        no2 = no2->esq;
    }
    return no1;
}

ArvAlbum *inserir_album(ArvAlbum *raiz, Album a, int *resp)
{
    if (raiz == NULL)
```

```
    {
        ArvAlbum *novo;
        novo = (ArvAlbum*)malloc(sizeof(ArvAlbum));
        if(novo == NULL)
        {
            *resp = 0;
            return NULL;
        }
        novo->album = a;
        novo->cor = RED;
        novo->dir = NULL;
        novo->esq = NULL;
        novo->musica = criarLista();
        *resp = 1;
        return novo;
    }
    if(strcmp(a.titulo, raiz->album.titulo) == 0)
        *resp = 0;
    else
    {
        if (a.ano < raiz->album.ano)
            raiz->esq = inserir_album(raiz->esq, a, resp);
        else
            raiz->dir = inserir_album(raiz->dir, a, resp);
    }
    if (cor_album(raiz->dir) == RED && cor_album(raiz->esq) == BLACK)
        rotacionaEsquerdaAlbum(&raiz);
    if (cor_album(raiz->esq) == RED && cor_album(raiz->esq->esq) == RED)
        rotacionaDireitaAlbum(&raiz);
    if(cor_album(raiz->dir) == RED && cor_album(raiz->esq) == RED)
        trocaCor_album(raiz);

    return raiz;

}

int insereArvLLRB_album(ArvAlbum **raiz, Album a)
{
    int resp;
    *raiz = inserir_album(*raiz, a, &resp);
    if (*raiz != NULL)
        (*raiz)->cor = BLACK;

    return resp;
}

ArvAlbum *excluir_album(ArvAlbum *raiz, char *titulo)
{
    if (raiz == NULL)
        return NULL;
```

```
    if (strcmp(titulo, raiz->album.titulo) < 0)
        raiz->esq = excluir_album(raiz->esq, titulo);
    else if (strcmp(titulo, raiz->album.titulo) > 0)
        raiz->dir = excluir_album(raiz->dir, titulo);
    else
    {

        if (raiz->esq == NULL && raiz->dir == NULL)
        { //sem filho

            free(raiz);
            return NULL;
        }
        else if (raiz->esq == NULL || raiz->dir == NULL)
        { //um filho

            ArvAlbum *temp = (raiz->esq != NULL) ? raiz->esq : raiz->dir;
            free(raiz);
            return temp;
        }
        else
        { //dois filhos
            ArvAlbum *temp = procuraMenorAlbum(raiz->dir);
            raiz->album = temp->album;
            raiz->dir = excluir_album(raiz->dir, temp->album.titulo);
        }

    }

    balancear_album(&raiz);

    return raiz;
}

ArvAlbum* buscaArvSAlbum(ArvAlbum *raiz, char *titulo, int *achou)
{
    if (raiz != NULL)
    {
        if (strcmp(titulo, raiz->album.titulo) < 0)
            return buscaArvSAlbum(raiz->esq, titulo, achou);
        else if (strcmp(titulo, raiz->album.titulo) > 0)
            return buscaArvSAlbum(raiz->dir, titulo, achou);
        else
        {
            *achou = 1;
            return raiz;
        }
    }

    *achou = 0;
    return NULL;
```

```c
}

void imprimir_um_album(Album a)
{
    printf("Ano: %d\nTitulo: %s\nQuantidade de musicas: %d\n\n", a.ano, a.titulo,
a.quantidade_musica);
}

void imprimir_todos_album(ArvAlbum *raiz)
{
    if (raiz != NULL)
    {
        imprimir_todos_album(raiz->esq);
        imprimir_um_album(raiz->album);
        imprimir_todos_album(raiz->dir);
    }

}

/*Musica*/

Musica *criarLista()
{
    return NULL;
}

void inserirMusica(ArvAlbum *lista)
{
    Musica *m = (Musica *)malloc(sizeof(Musica));
    *m = ler_musica();

    Musica *atual = lista->musica;
    Musica *anterior = NULL;


    while (atual != NULL && strcmp(m->nome_musica, atual->nome_musica) > 0)
    {
        anterior = atual;
        atual = atual->prox;
    }
    if (anterior == NULL)
    {
        m->prox = lista->musica;
        lista->musica = m;
    }
    else
    {
        anterior->prox = m;
        m->prox = atual;
    }
    lista->album.quantidade_musica++;
```

```c
}


Musica *busca_musica(ArvAlbum *lista, char *nome)
{
    Musica *atual = lista->musica;

    while (atual != NULL && strcmp(atual->nome_musica, nome) != 0)
        atual = atual->prox;

    return atual;
}

void excluir_musica(ArvAlbum *lista, char *nome, int *excluiu)
{
    Musica *anterior = NULL;
    Musica *atual = lista->musica;

    while (atual != NULL && strcmp(atual->nome_musica, nome) != 0)
    {
        anterior = atual;
        atual = atual->prox;
    }
    if (atual != NULL)
    {
        if (anterior == NULL)
            lista->musica = atual->prox;
        else
            anterior->prox = atual->prox;
        free(atual);
        *excluiu = 1;
    }else
        *excluiu = 0;
}

void imprimir_todas_musicas(ArvAlbum *lista)
{
    Musica *m = lista->musica;
    while (m != NULL)
    {
        imprimir_uma_musica(m);
        m = m->prox;
    }
}

void imprimir_uma_musica(Musica* m)
{
    printf("Nome da musica: %s\nDuracao em minutos: %d min\n", m->nome_musica, m-
>minutos);
}
```

```c
void liberarMusicas(Musica *lista) {
    while (lista != NULL) {
        Musica *temp = lista;
        lista = lista->prox;
        free(temp);
    }
}

void liberarAlbums(ArvAlbum *raiz) {
    if (raiz != NULL) {
        liberarAlbums(raiz->esq);
        liberarAlbums(raiz->dir);
        liberarMusicas(raiz->musica);
        free(raiz);
    }
}

void liberarArtistas(ArvArtista *raiz) {
    if (raiz != NULL) {
        liberarArtistas(raiz->esq);
        liberarArtistas(raiz->dir);
        liberarAlbums(raiz->arvAlbum);
        free(raiz);
    }
}

void liberarTodaMemoria(ArvArtista *raiz) {
    liberarArtistas(raiz);
}

void menu()
{
    printf("==============MENU==============\n");
    printf(": 1 - Cadastro de Artista      :\n");
    printf(": 2 - Excluir Artista          :\n");
    printf(": 3 - Imprimir Artistas        :\n");
    printf(": 4 - Busca                    :\n");
    printf(": 5 - Cadastro de Album        :\n");
    printf(": 6 - Excluir Album            :\n");
    printf(": 7 - Imprimir Albuns          :\n");
    printf(": 8 - Cadastro de Musica       :\n");
    printf(": 9 - Excluir Musica           :\n");
    printf(": 10 - Imprimir Musica         :\n");
    printf(": 0 - Sair                     :\n");
    printf("================================\n");
}

void menu_busca()
{
    system("cls");
    printf("==============MENU==============\n");
```

```
    printf(": 1 - Buscar Artista           :\n");
    printf(": 2 - Buscar Album             :\n");
    printf(": 3 - Buscar Musica            :\n");
    printf(": 0 - Sair                     :\n");
    printf("================================\n");
}



int main()
{
    ArvArtista *raiz = NULL;

    int op = -1;

    while (op != 0)
    {
        menu();
        scanf("%d", &op);
        switch (op)
        {
            case 1:
            {
                int result = insereArvLLRB_artista(&raiz, ler_artista());
                if (result == 1)
                    printf("Artista inserido.\n");
                else
                    printf("O artista ja esta cadastrado.\n");
                break;
            }

            case 2:
            {
                fflush(stdin);
                int op_excluir;
                printf("Todos os albuns e musicas do artista serao excluidos
juntamente. Continuar? \n1 - Sim\n2 - Nao\n");
                scanf("%d", &op_excluir);

                if (op_excluir == 1)
                {
                    fflush(stdin);
                    char nome[50];
                    fflush(stdin);
                    printf("Informe o artista a ser excluido: ");
                    scanf("%[^\n]", nome);
                    raiz = excluir_artista(raiz, nome);
                    if (raiz != NULL)
                        printf("Artista excluido com sucesso!!\n");
                    else
                        printf("Artista nao encontrado!!\n");
```

```
            }else if(op_excluir == 2)
                printf("Operacao cancelada!!\n");

            break;
        }

        case 3:
            imprimir_todos_artistas(raiz);
            break;
        case 4:
        {
            int op_busca;
            menu_busca();
            scanf("%d", &op_busca);
            if (op_busca == 1)
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista a ser buscado: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                    imprimir_um(busca->artista);
                else
                    printf("Artista nao encontrado!!\n");
            }
            else if(op_busca == 2)
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o album a ser buscado: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvAlbum *busca = buscaArvSAlbum(raiz->arvAlbum, nome, &achou);
                if (achou == 1)
                    imprimir_um_album(busca->album);
                else
                    printf("Album nao encontrado!!\n");
            }
            else if (op_busca == 3)
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja mostrar a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
```

```
                    if (achou == 1)
                    {
                        achou = 0;
                        char nome_album[50];
                        fflush(stdin);
                        printf("Informe o album que deseja mostrar a musica: ");
                        scanf("%[^\n]", nome_album);
                        ArvAlbum *busca_album  = buscaArvSAlbum(busca->arvAlbum,
nome_album, &achou);
                        if (achou == 1)
                        {
                            char nome_musica[50];
                            printf("Informe o nome da musica: ");
                            scanf("%[^\n]", nome_musica);
                            Musica *mus = busca_musica(busca_album, nome_musica);
                            if (mus != NULL)
                                imprimir_uma_musica(mus);
                            else
                                printf("Musica nao encontrada!!\n");

                        }
                        else
                            printf("Album nao encontrado!!\n");

                    }
                    else
                        printf("Artista nao encontrado!!\n");
                }
                break;
            }

            case 5:
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja cadastrar um album: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                {

                    int result = insereArvLLRB_album(&busca->arvAlbum,
ler_album());

                    if (result == 1)
                    {
                        busca->artista.num_albuns++;
                        printf("Album inserido.\n");
                    }
```

```
                    else
                        printf("O album ja esta cadastrado.\n");
                }
                else
                    printf("Artista nao encontrado!!\n");

                break;
            }

            case 6:
            {
                fflush(stdin);
                int op_excluir;
                printf("Todos as musicas do album serao excluidos juntamente.
Continuar? \n1 - Sim\n2 - Nao\n");
                scanf("%d", &op_excluir);
                if (op_excluir == 1)
                {
                    int achou = 0;
                    char nome[50];
                    fflush(stdin);
                    printf("Informe o artista que deseja excluir o album: ");
                    scanf("%[^\n]", nome);
                    fflush(stdin);
                    ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                    if (achou == 1)
                    {
                        char titulo[50];
                        fflush(stdin);
                        printf("Informe o album a ser excluido: ");
                        scanf("%[^\n]", titulo);
                        fflush(stdin);
                        busca->arvAlbum = excluir_album(busca->arvAlbum, titulo);
                        if (busca->arvAlbum != NULL)
                        {
                            busca->artista.num_albuns--;
                            printf("Album excluido com sucesso!!\n");
                        }
                        else
                            printf("Album nao encontrado!!\n");
                    }
                    else
                    printf("Artista nao encontrado!!\n");

                }else
                    printf("Operacao cancelada!!\n");

                break;
            }

            case 7:
```

```
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja mostrar os albuns: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                    imprimir_todos_album(busca->arvAlbum);
                else
                    printf("Artista nao encontrado!!\n");

                break;
            }
            case 8:
            {
                fflush(stdin);
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja cadastrar a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                {
                    achou = 0;
                    char nome_album[50];
                    fflush(stdin);
                    printf("Informe o album que deseja cadastrar musica: ");
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvSAlbum(busca->arvAlbum,
nome_album, &achou);
                    if (achou == 1)
                    {
                        inserirMusica(busca_album);
                        printf("Musica inserida com sucesso!!\n");
                    }
                    else
                        printf("Album nao encontrado!!\n");

                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            case 9:
            {
                int achou = 0;
                char nome[50];
```

```
                fflush(stdin);
                printf("Informe o artista que deseja excluir a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                {
                    achou = 0;
                    char nome_album[50];
                    fflush(stdin);
                    printf("Informe o album que deseja excluir a musica: ");
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvSAlbum(busca->arvAlbum,
nome_album, &achou);
                    if (achou == 1)
                    {
                        char nome_musica[50];
                        int excluiu = 0;
                        fflush(stdin);
                        printf("Informe a musica que deseja excluir: ");
                        scanf("%[^\n]", nome_musica);
                        excluir_musica(busca_album, nome_musica, &excluiu);
                        if (excluiu == 1)
                        {
                            busca_album->album.quantidade_musica--;
                            printf("Musica deletada com sucesso!!\n");
                        }
                        else
                            printf("Musica nao encontrada!!\n");
                    }
                    else
                        printf("Album nao encontrado!!\n");

                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            case 10:
            {
                int achou = 0;
                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja mostrar a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvSArtista(raiz, nome, &achou);
                if (achou == 1)
                {
                    achou = 0;
                    char nome_album[50];
```

```
                    fflush(stdin);
                    printf("Informe o album que deseja mostrar as musicas: ");
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvSAlbum(busca->arvAlbum,
nome_album, &achou);
                    if (achou == 1)
                        imprimir_todas_musicas(busca_album);
                    else
                        printf("Album nao encontrado!!\n");

                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            default:
                break;
        }
    }

    liberarTodaMemoria(raiz);
    return 0;
}
```

Questão 1.1:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <windows.h>


#define RED 1
#define BLACK 0

typedef struct artista
{
    char nome[50];
}Artista;

typedef struct arvartista
{
    Artista artista;
    struct arvartista *esq, *dir;
    int cor;
}ArvArtista;

int cor_artista(ArvArtista *raiz)
{
    if (raiz == NULL)
        return BLACK;
```

```
    else
        return raiz->cor;
}

void trocaCor_artista(ArvArtista *raiz)
{
    raiz->cor = !raiz->cor;
    if (raiz->esq != NULL)
        raiz->esq->cor = !raiz->esq->cor;
    if(raiz->dir != NULL)
        raiz->dir->cor = !raiz->dir->cor;
}

void rotacionaEsquerdaArtista(ArvArtista **A)
{
    ArvArtista *B = (*A)->dir;
    (*A)->dir = B->esq;
    B->esq = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}

void rotacionaDireitaArtista(ArvArtista **A)
{
    ArvArtista *B = (*A)->esq;
    (*A)->esq = B->dir;
    B->dir = *A;
    B->cor = (*A)->cor;
    (*A)->cor = RED;
    *A = B;
}

void move2EsqREDArtista(ArvArtista **raiz)
{
    trocaCor_artista(*raiz);
    if (cor_artista((*raiz)->dir->esq) == RED)
    {
        rotacionaDireitaArtista(&(*raiz)->dir);
        rotacionaEsquerdaArtista(&(*raiz));
        trocaCor_artista(*raiz);
    }
}

void move2DirREDArtista(ArvArtista **raiz)
{
    trocaCor_artista(*raiz);
    if (cor_artista((*raiz)->esq->esq) == RED)
    {
        rotacionaDireitaArtista(&(*raiz));
        trocaCor_artista(*raiz);
```

```
    }
}

void balancear_artista(ArvArtista **raiz)
{
    if (cor_artista((*raiz)->dir) == RED)
        rotacionaEsquerdaArtista(raiz);
    if((*raiz)->esq != NULL && cor_artista((*raiz)->esq) == RED &&
cor_artista((*raiz)->esq->esq) == RED)
        rotacionaDireitaArtista(raiz);
    if (cor_artista((*raiz)->esq) == RED && cor_artista((*raiz)->dir) == RED)
        trocaCor_artista(*raiz);

}

ArvArtista *removerMenor(ArvArtista *raiz)
{
    if ((raiz->esq) == NULL)
    {
        free(raiz);
        return raiz;
    }
    if (cor_artista(raiz->esq) == BLACK && cor_artista(raiz->esq->esq) == BLACK)
        move2EsqREDArtista(&raiz);

    removerMenor(raiz->esq);

    balancear_artista(&raiz);

    return raiz;
}

ArvArtista *procuraMenor(ArvArtista *atual)
{
    ArvArtista *no1 = atual;
    ArvArtista *no2 = atual->esq;
    while (no2 != NULL)
    {
        no1 = no2;
        no2 = no2->esq;
    }
    return no1;
}

ArvArtista *inserir_artista(ArvArtista *raiz, Artista a, int *resp)
{
    if (raiz == NULL)
    {
        ArvArtista *novo;
        novo = (ArvArtista*)malloc(sizeof(ArvArtista));
        if(novo == NULL)
```

```
        {
            *resp = 0;
            return NULL;
        }
        novo->artista = a;
        novo->cor = RED;
        novo->dir = NULL;
        novo->esq = NULL;
        *resp = 1;
        return novo;
    }
    if(strcmp(a.nome, raiz->artista.nome) == 0)
        *resp = 0;
    else
    {
        if (strcmp(a.nome, raiz->artista.nome) < 0)
            raiz->esq = inserir_artista(raiz->esq, a, resp);
        else
            raiz->dir = inserir_artista(raiz->dir, a, resp);
    }
    if (cor_artista(raiz->dir) == RED && cor_artista(raiz->esq) == BLACK)
        rotacionaEsquerdaArtista(&raiz);
    if (cor_artista(raiz->esq) == RED && cor_artista(raiz->esq->esq) == RED)
        rotacionaDireitaArtista(&raiz);
    if(cor_artista(raiz->dir) == RED && cor_artista(raiz->esq) == RED)
        trocaCor_artista(raiz);

    return raiz;

}

ArvArtista **buscaTestes(ArvArtista **tree, char *codigo)
{
    ArvArtista **NO;
    NO = NULL;
    if (*tree != NULL)
    {
        if (strcmp(codigo, (*tree)->artista.nome) == 0)
        {
            NO = tree;
        }
        else
        {
            if (strcmp(codigo, (*tree)->artista.nome) < 0)
            {
                NO = buscaTestes(&(*tree)->esq, codigo);
            }
            else
            {
                NO = buscaTestes(&(*tree)->dir, codigo);
            }
```

```
        }
    }
    return NO;
}

int main()
{

    ArvArtista *tree = NULL;
    double elapsed_nanos;
    double soma = 0;




    char *artistas[30] = {
        "Ana", "Beatriz", "Carlos", "Daniel", "Eduardo", "Fernanda", "Gabriel",
"Helena", "Igor",
        "Julia", "Kleber", "Laura", "Marcos", "Natalia", "Otavio", "Patricia",
"Quirino", "Rafael",
        "Sabrina", "Tiago", "Ursula", "Vitor", "Wagner", "Ximena", "Yasmin",
"Zacarias",
        "Alice", "Bruno", "Cecilia", "Diego"};


    for (int i = 0; i < 30; i++)
    {
        Artista artista1;
        strncpy(artista1.nome, artistas[i], sizeof(artista1.nome) - 1);
        artista1.nome[sizeof(artista1.nome) - 1] = '\0';


        int resp = 0;
        tree = inserir_artista(tree, artista1, &resp);

        LARGE_INTEGER start, end, frequency;

        QueryPerformanceFrequency(&frequency);
        QueryPerformanceCounter(&start);


        buscaTestes(&tree, artistas[i]);

        QueryPerformanceCounter(&end);


        elapsed_nanos = ((end.QuadPart - start.QuadPart) * 1.0 /
frequency.QuadPart) * 1000000000;
        //printf("Tempo de execucao: %.2f nanossegundos\n", elapsed_nanos);
        soma += elapsed_nanos;
```

```
    }

    printf("Tempo total de execucao: %.2f nanossegundos\n", soma);
    printf("Media da soma total de execucao: %.2f nanossegundos\n", soma/30);

    return 0;
}
```

Questão 2:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Prototypes */

typedef struct artista Artista;
typedef struct album Album;
typedef struct arvartista ArvArtista;
typedef struct arvalbum ArvAlbum;
/* Structs infos */

typedef struct artista
{
    char nome_artista[50];
    char estilo[50];
    int num_albuns;

} Artista;

typedef struct album
{
    char titulo[50];
    int ano;
    int quantidade_musica;
}Album;

typedef struct musica
{
    char nome_musica[50];
    int minutos;
    struct musica *prox;
}Musica;


/* Structs arvores */

typedef struct arvartista
{
    Artista info_1, info_2;
    int Nkeys;
```

```
    ArvAlbum *arvAlbum;
    struct arvartista *esq, *cen, *dir;
} ArvArtista;

typedef struct arvalbum
{
    Album info_1, info_2;
    Musica *musica;
    int Nkeys;
    struct arvalbum *esq, *cen,*dir;
}ArvAlbum;

/* Function Declarations */

Artista ler_artista();
ArvArtista *criaArvArtista();
int ehFolhaArtista(ArvArtista *raiz);
void criaNoArtista(ArvArtista **no, Artista a, ArvArtista *esq, ArvArtista *cen);
void adicionaNoArtista(ArvArtista **no, Artista info, ArvArtista *filho);
void quebraNoArtista(ArvArtista **no, Artista info, Artista *sobe, ArvArtista
**no_maior, ArvArtista *filho);
ArvArtista *insereArvArtista(ArvArtista **pai, ArvArtista **raiz, Artista info,
Artista *sobe);
void imprimirArvArtista(ArvArtista *raiz);
void liberaNoArtista(ArvArtista **no);
int removeEsqFolhaArtista(ArvArtista **raiz, ArvArtista **pai);
int removeCenFolhaArtista(ArvArtista **raiz, ArvArtista **pai);
void removeDirFolhaArtista(ArvArtista **raiz, ArvArtista **pai);
int removeFolhaArtista(ArvArtista **pai, ArvArtista **raiz, char *nome);
int removeNoFilhoFolhaArtista(ArvArtista **raiz, char *nome);
int balanceiaArvArtista(ArvArtista **raiz, int desbalanceamento);
int removeMenorNoArtista(ArvArtista **pai_aux, ArvArtista **no, Artista
*info_sobe);
int removerArtista(ArvArtista **pai, ArvArtista **raiz, char *nome);
ArvArtista *buscaArvArtista(ArvArtista **raiz, char *nome);

Album ler_album();
ArvAlbum *criaArvAlbum();
int ehFolhaAlbum(ArvAlbum *raiz);
void criaNoAlbum(ArvAlbum **no, Album a, ArvAlbum *esq, ArvAlbum *cen);
void adicionaNoAlbum(ArvAlbum **no, Album info, ArvAlbum *filho);
void quebraNoAlbum(ArvAlbum **no, Album info, Album *sobe, ArvAlbum **no_maior,
ArvAlbum *filho);
ArvAlbum *insereArvAlbum(ArvAlbum **pai, ArvAlbum **raiz, Album info, Album *sobe);
void imprimirArvAlbum(ArvAlbum *raiz);
void liberaNoAlbum(ArvAlbum **no);
int removeEsqFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai);
int removeCenFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai);
void removeDirFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai);
int removeFolhaAlbum(ArvAlbum **pai, ArvAlbum **raiz, char *nome);
int removeNoFilhoFolhaAlbum(ArvAlbum **raiz, char *nome);
```

```c
int balanceiaArvAlbum(ArvAlbum **raiz, int desbalanceamento);
int removeMenorNoAlbum(ArvAlbum **pai_aux, ArvAlbum **no, Album *info_sobe);
int removerAlbum(ArvAlbum **pai, ArvAlbum **raiz, char *nome);
ArvAlbum *buscaArvAlbum(ArvAlbum **raiz, char *nome);
Musica ler_musica();
Musica *criarLista();
void inserirMusica(ArvAlbum *lista);
Musica *busca_musica(ArvAlbum *lista, char *nome);
void excluir_musica(ArvAlbum *lista, char *nome);
void imprimir_todas_musicas(ArvAlbum *lista);
void imprimir_uma_musica(Musica* m);

/* Function Definitions */

Artista ler_artista()
{
    Artista art;

    fflush(stdin);
    printf("Informe o nome do artista: ");
    scanf(" %[^\n]s", art.nome_artista);
    fflush(stdin);
    printf("Informe o estilo musical: ");
    scanf(" %[^\n]s", art.estilo);
    fflush(stdin);
    art.num_albuns = 0;
    fflush(stdin);
    return art;
}

ArvArtista *criaArvArtista()
{
    return NULL;
}

int ehFolhaArtista(ArvArtista *raiz)
{
    int folha = 0;
    if (raiz != NULL)
        if (raiz->esq == NULL)
            folha = 1;
    return folha;
}

void criaNoArtista(ArvArtista **no, Artista a, ArvArtista *esq, ArvArtista *cen)
{
    *no = (ArvArtista*)malloc(sizeof(ArvArtista));

    (*no)->info_1 = a;
    (*no)->arvAlbum = criaArvAlbum();
    (*no)->Nkeys = 1;
```

```
    (*no)->esq = esq;
    (*no)->cen = cen;
    (*no)->dir = NULL;
}

void adicionaNoArtista(ArvArtista **no, Artista info, ArvArtista *filho)
{
    if (strcmp(info.nome_artista, (*no)->info_1.nome_artista) > 0)
    {
        (*no)->info_2 = info;
        (*no)->dir = filho;
    }else{
        (*no)->info_2 = (*no)->info_1;
        (*no)->info_1 = info;
        (*no)->dir = (*no)->cen;
        (*no)->cen = filho;
    }
    (*no)->Nkeys = 2;
}

void quebraNoArtista(ArvArtista **no, Artista info, Artista *sobe, ArvArtista
**no_maior, ArvArtista *filho)
{
    if (strcmp(info.nome_artista, (*no)->info_1.nome_artista) < 0)
    {
        *sobe = (*no)->info_1;
        (*no)->info_1 = info;
        criaNoArtista(no_maior, (*no)->info_2, (*no)->cen, (*no)->dir);
        (*no)->cen = filho;
    }else if (strcmp(info.nome_artista, (*no)->info_2.nome_artista) < 0)
    {
       *sobe = info;
       criaNoArtista(no_maior, (*no)->info_2, filho, (*no)->dir);
    }else{
        *sobe = (*no)->info_2;
        criaNoArtista(no_maior, info, (*no)->dir, filho);
    }
    (*no)->dir = NULL;
    (*no)->Nkeys = 1;

}

ArvArtista *insereArvArtista(ArvArtista **pai, ArvArtista **raiz, Artista info,
Artista *sobe)
{
    ArvArtista *no_maior = NULL;
    if (*raiz == NULL)
        criaNoArtista(raiz, info, NULL, NULL);
    else{
        if (ehFolhaArtista(*raiz) == 1)
        {
```

```
            if ((*raiz)->Nkeys == 1)
                adicionaNoArtista(raiz, info, NULL);
            else{
                quebraNoArtista(raiz, info, sobe, &no_maior, NULL);
                if (*pai == NULL)
                {
                    ArvArtista *aux;
                    criaNoArtista(&aux, *sobe, *raiz, no_maior);
                    *raiz = aux;
                    no_maior = NULL;
                }
            }
        }else{
            if (strcmp(info.nome_artista, (*raiz)->info_1.nome_artista) < 0)
                no_maior = insereArvArtista(raiz, &(*raiz)->esq, info, sobe);
            else if(strcmp(info.nome_artista, (*raiz)->info_2.nome_artista) > 0 &&
(*raiz)->Nkeys == 2)
                no_maior = insereArvArtista(raiz, &(*raiz)->dir, info, sobe);
            else
                no_maior = insereArvArtista(raiz, &(*raiz)->cen, info, sobe);
            if (no_maior != NULL)
            {
                if ((*raiz)->Nkeys == 1)
                {
                    adicionaNoArtista(raiz, *sobe, no_maior);
                    no_maior = NULL;
                }else{
                    Artista sobe2;
                    ArvArtista *no_maior_2;

                    quebraNoArtista(raiz, *sobe, &sobe2, &no_maior_2, no_maior);
                    if (*pai == NULL)
                    {
                        ArvArtista *aux;
                        criaNoArtista(&aux, sobe2, *raiz, no_maior_2);
                        *raiz = aux;
                        no_maior = NULL;
                    }else{
                        *sobe = sobe2;
                        no_maior = no_maior_2;
                    }

                }

            }

        }
    }
    return no_maior;
}
```

```c
void imprimirArvArtista(ArvArtista *raiz)
{
    if (raiz != NULL)
    {
        imprimirArvArtista(raiz->esq);
        if (raiz->Nkeys == 1)
            printf("Nome do artista: %s\nEstilo musical: %s\nQuantidade de albuns:
%d\n", raiz->info_1.nome_artista, raiz->info_1.estilo, raiz->info_1.num_albuns);
        else if (raiz->Nkeys == 2)
        {
            printf("Nome do artista: %s\nEstilo musical: %s\nQuantidade de albuns:
%d\n", raiz->info_1.nome_artista, raiz->info_1.estilo, raiz->info_1.num_albuns);
            printf("Nome do artista: %s\nEstilo musical: %s\nQuantidade de albuns:
%d\n", raiz->info_2.nome_artista, raiz->info_2.estilo, raiz->info_2.num_albuns);
        }
        imprimirArvArtista(raiz->cen);
        imprimirArvArtista(raiz->dir);
    }
}
/*-----------------------------------------------------------------*/

void liberaNoArtista(ArvArtista **no)
{
    ArvArtista *aux;

    aux = *no;
    *no = NULL;
    free(aux);
}

int removeEsqFolhaArtista(ArvArtista **raiz, ArvArtista **pai)
{
    int flag = 0;
    (**raiz).info_1 = (**pai).info_1;

    if ((**pai).cen->Nkeys == 2)
    {
        (**pai).info_1 = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).cen->info_2;
        (**pai).cen->Nkeys = 1;
    }else if((**pai).Nkeys == 2){
        (**pai).info_1 = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).info_2;
        if ((**pai).dir->Nkeys == 2)
        {
            (**pai).info_2 = (**pai).dir->info_1;
            (**pai).dir->info_1 = (**pai).dir->info_2;
            (**pai).dir->Nkeys = 1;
        }
        else{
            adicionaNoArtista(&(**pai).cen, (**pai).dir->info_1, NULL);
```

```
            liberaNoArtista(&(**pai).dir);
            (**pai).Nkeys = 1;
        }

    }else{
        adicionaNoArtista(pai, (**pai).cen->info_1, NULL);
        (**pai).Nkeys = 2;
        liberaNoArtista(&(**pai).esq);
        liberaNoArtista(&(**pai).cen);
        flag = 1;
    }

    return flag;
}

int removeCenFolhaArtista(ArvArtista **raiz, ArvArtista **pai)
{
    int flag = 0;

    if ((**pai).esq->Nkeys == 2)
    {
        (**raiz).info_1 = (**pai).info_1;
        (**pai).info_1 = (**pai).esq->info_2;
        (**pai).esq->Nkeys = 1;
    }else if ((**pai).Nkeys == 2)
    {
        (**raiz).info_1 = (**pai).info_2;
        if ((**pai).dir->Nkeys == 2)
        {
            (**pai).info_2 = (**pai).dir->info_1;
            (**pai).dir->info_1 = (**pai).dir->info_2;
            (**pai).dir->Nkeys = 1;
        }else{
            adicionaNoArtista(&(**pai).cen, (**pai).dir->info_1, NULL);
            liberaNoArtista(&(**pai).dir);
            (**pai).Nkeys = 1;
        }
    }else{
        adicionaNoArtista(pai, (**pai).esq->info_1, NULL);
        (**pai).Nkeys = 2;
        liberaNoArtista(&(**pai).esq);
        liberaNoArtista(&(**pai).dir);
        flag = 1;
    }

    return flag;
}

void removeDirFolhaArtista(ArvArtista **raiz, ArvArtista **pai)
{
    (**raiz).info_1 = (**pai).info_2;
```

```
    if ((**pai).cen->Nkeys == 2)
    {
        (**pai).info_2 = (**pai).cen->info_2;
        (**pai).cen->Nkeys = 1;
    }else if ((**pai).esq->Nkeys == 2)
    {
        Artista aux = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).info_1;
        (**pai).info_1 = (**pai).esq->info_2;
        (**pai).info_2 = aux;
        (**pai).esq->Nkeys = 1;
    }else{
        adicionaNoArtista(&(**pai).cen, (**pai).info_2, NULL);
        (**pai).Nkeys = 1;
        liberaNoArtista(&(**pai).dir);
    }
}

int removeFolhaArtista(ArvArtista **pai, ArvArtista **raiz, char *nome)
{
    int balanceamento = 0;
    if ((*raiz)->Nkeys == 2)
    {
        if (strcmp(nome, (*raiz)->info_1.nome_artista) == 0)
            (*raiz)->info_1 = (*raiz)->info_2;
        (*raiz)->Nkeys = 1;
    }else if (*pai == NULL)
    {
        liberaNoArtista(raiz);
    }else{
        if (*raiz == (**pai).esq)
            balanceamento = removeEsqFolhaArtista(raiz, pai);
        else if (*raiz == (**pai).cen)
            balanceamento = removeCenFolhaArtista(raiz, pai);
        else if (*raiz == (**pai).dir)
            removeDirFolhaArtista(raiz, pai);
    }

    return balanceamento;
}

int removeNoFilhoFolhaArtista(ArvArtista **raiz, char *nome)
{
    int balanceamento = 0;
    if (strcmp(nome, (**raiz).info_1.nome_artista) == 0)
    {
        if ((**raiz).cen->Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).cen->info_2;
            (**raiz).Nkeys = 1;
```

```
        }else if ((**raiz).esq->Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).esq->info_2;
            (**raiz).esq->Nkeys = 1;
        }else if ((**raiz).Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).info_2;
            if ((**raiz).dir->Nkeys == 2)
            {
                (**raiz).info_2 = (**raiz).dir->info_1;
                (**raiz).dir->info_1 = (**raiz).dir->info_2;
                (**raiz).dir->Nkeys = 1;
            }else{
                adicionaNoArtista(&(**raiz).cen, (**raiz).dir->info_1, NULL);
                (**raiz).Nkeys = 1;
                liberaNoArtista(&(**raiz).dir);
            }

        }else{
            ArvArtista *aux;
            adicionaNoArtista(&(**raiz).esq, (**raiz).cen->info_1, NULL);
            aux = (**raiz).esq;
            liberaNoArtista(&(**raiz).cen);
            liberaNoArtista(raiz);
            *raiz = aux;
            balanceamento = 1;
        }
    }else{
        if ((**raiz).dir->Nkeys == 2)
        {
            (**raiz).info_2 = (**raiz).dir->info_1;
            (**raiz).dir->info_1 = (**raiz).dir->info_2;
            (**raiz).dir->Nkeys = 1;
        }else if ((**raiz).cen->Nkeys == 2)
        {
            (**raiz).info_2 = (**raiz).cen->info_2;
            (**raiz).cen->Nkeys = 1;
        }else if ((**raiz).esq->Nkeys == 2)
        {
            (**raiz).info_2 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).info_1;
            (**raiz).info_1 = (**raiz).esq->info_2;
            (**raiz).esq->Nkeys = 1;
        }else{
            adicionaNoArtista(&(**raiz).cen, (**raiz).dir->info_1, NULL);
            (**raiz).Nkeys = 1;
            liberaNoArtista(&(**raiz).dir);
        }
    }
```

```
    return balanceamento;
}

int balanceiaArvArtista(ArvArtista **raiz, int desbalanceamento)
{
    Artista sobe;
    ArvArtista *no_maior;
    ArvArtista *aux;

    if (desbalanceamento == -1)
    {
        aux = (*raiz)->cen->esq;
        (*raiz)->cen->esq = (*raiz)->esq;
        if ((*raiz)->cen->Nkeys == 1)
        {
            adicionaNoArtista(&(*raiz)->cen, (*raiz)->info_1, aux);
            (*raiz)->esq = NULL;
            if ((*raiz)->Nkeys == 1)
            {
                aux = *raiz;
                *raiz = (*raiz)->cen;
                liberaNoArtista(&aux);
            }else{
                (*raiz)->info_1 = (*raiz)->info_2;
                (*raiz)->esq = (*raiz)->cen;
                (*raiz)->cen = (*raiz)->dir;
                (*raiz)->dir = NULL;
                (*raiz)->Nkeys = 1;
            }

        }else{
            quebraNoArtista(&(*raiz)->cen, (*raiz)->info_1, &sobe, &no_maior, aux);
            (*raiz)->info_1 = sobe;
            (*raiz)->esq = (*raiz)->cen;
            (*raiz)->cen = no_maior;
        }
    }else if (desbalanceamento == 0)
    {
        if ((*raiz)->esq->Nkeys == 1)
        {
            adicionaNoArtista(&(*raiz)->esq, (*raiz)->info_1, (*raiz)->cen);
            (*raiz)->cen = NULL;
            if ((*raiz)->Nkeys == 1)
            {
                aux = *raiz;
                *raiz = (*raiz)->esq;
                liberaNoArtista(&aux);
            }else{
                (*raiz)->info_1 = (*raiz)->info_2;
                (*raiz)->cen = (*raiz)->dir;
                (*raiz)->dir = NULL;
```

```
                (*raiz)->Nkeys = 1;
            }

        }else{
            quebraNoArtista(&(*raiz)->esq, (*raiz)->info_1, &sobe, &no_maior,
(*raiz)->cen);
            (*raiz)->info_1 = sobe;
            (*raiz)->cen = no_maior;
        }
    }else{
        if ((*raiz)->cen->Nkeys == 1)
        {
            adicionaNoArtista(&(*raiz)->cen, (*raiz)->info_2, (*raiz)->dir);
            (*raiz)->Nkeys = 1;
            (*raiz)->dir = NULL;
        }else{
            quebraNoArtista(&(*raiz)->cen, (*raiz)->info_2, &sobe, &no_maior,
(*raiz)->dir);
            (*raiz)->info_2 = sobe;
            (*raiz)->dir = no_maior;
        }
    }

    return 0;
}

int removeMenorNoArtista(ArvArtista **pai_aux, ArvArtista **no, Artista *info_sobe)
{
    int balanceamento = 0;
    if ((*no)->esq != NULL)
        balanceamento = removeMenorNoArtista(no, &(*no)->esq, info_sobe);
    else{
        *info_sobe = (*no)->info_1;
        balanceamento = removeFolhaArtista(pai_aux, no, (*no)-
>info_1.nome_artista);
    }
    return balanceamento;
}

int removerArtista(ArvArtista **pai, ArvArtista **raiz, char *nome)
{
    int balanceamento = 0;
    if (*raiz != NULL)
    {
        if ((strcmp(nome, (*raiz)->info_1.nome_artista) == 0) || ((*raiz)->Nkeys ==
2 && strcmp(nome, (*raiz)->info_2.nome_artista) == 0))
        {

            if (ehFolhaArtista(*raiz) == 1)
                balanceamento = removeFolhaArtista(pai, raiz, nome);
```

```
            else if (((ehFolhaArtista((*raiz)->esq) == 1) &&
ehFolhaArtista((*raiz)->cen) == 1 && ehFolhaArtista((*raiz)->dir) == 1) ||
((ehFolhaArtista((*raiz)->esq) == 1) && ehFolhaArtista((*raiz)->cen) == 1 &&
(*raiz)->Nkeys == 1 ))
                balanceamento = removeNoFilhoFolhaArtista(raiz, nome);
            else{
                ArvArtista *pai_aux = NULL;
                Artista info_sobe;
                if (strcmp(nome, (*raiz)->info_1.nome_artista) == 0)
                {
                    balanceamento = removeMenorNoArtista(&pai_aux, &(*raiz)->cen,
&info_sobe);
                    (*raiz)->info_1 = info_sobe;
                    if (balanceamento == 1)
                    {
                        balanceamento = balanceiaArvArtista(raiz, 0);
                    }
                }else{
                    balanceamento = removeMenorNoArtista(&pai_aux, &(*raiz)->dir,
&info_sobe);
                    (*raiz)->info_2 = info_sobe;
                    if (balanceamento == 1)
                    {
                        balanceamento = balanceiaArvArtista(raiz, 1);
                    }
                }
            }
        }
        else if (strcmp(nome,(*raiz)->info_1.nome_artista) < 0)
            balanceamento = removerArtista(raiz, &(*raiz)->esq, nome);
        else if (strcmp(nome,(*raiz)->info_2.nome_artista) > 0 && (*raiz)->Nkeys ==
2)
            balanceamento = removerArtista(raiz, &(*raiz)->dir, nome);
        else
            balanceamento = removerArtista(raiz, &(*raiz)->cen, nome);
        if (balanceamento == 1 && *raiz != NULL && *pai != NULL)
        {
            if (*raiz == (**pai).esq)
                balanceamento = balanceiaArvArtista(pai, -1);
            else if(*raiz == (*pai)->cen)
                balanceamento = balanceiaArvArtista(pai, 0);
            else if(*raiz == (*pai)->dir)
                balanceamento = balanceiaArvArtista(pai, 1);

        }

    }
    return balanceamento;
}

ArvArtista *buscaArvArtista(ArvArtista **raiz, char *nome)
```

```c
{
    ArvArtista *no = NULL;
    if (*raiz != NULL)
    {
        if (strcmp(nome, (*raiz)->info_1.nome_artista) == 0)
            no = *raiz;
        else if(strcmp(nome, (*raiz)->info_2.nome_artista) == 0)
            no = *raiz;
        else
        {
            if (strcmp(nome, (*raiz)->info_1.nome_artista) < 0)
                no = buscaArvArtista(&(*raiz)->esq, nome);
            else if (strcmp(nome, (*raiz)->info_2.nome_artista) < 0 || (*raiz)->Nkeys == 1)
                no = buscaArvArtista(&(*raiz)->cen, nome);
            else
                no = buscaArvArtista(&(*raiz)->dir, nome);
        }
    }

    return no;
}



/*-------------------------------------------------------------------------------*/

Album ler_album()
{
    Album album;
    fflush(stdin);
    printf("Informe o nome do album: ");
    scanf("%[^\n]s", album.titulo);
    fflush(stdin);
    printf("Informe o ano do album: ");
    scanf("%d", &album.ano);
    fflush(stdin);
    album.quantidade_musica = 0;

    return album;
}

ArvAlbum *criaArvAlbum()
{
    return NULL;
}

int ehFolhaAlbum(ArvAlbum *raiz)
{
    int folha = 0;
    if (raiz != NULL)
```

```
        if (raiz->esq == NULL)
            folha = 1;
    return folha;
}

void criaNoAlbum(ArvAlbum **no, Album a, ArvAlbum *esq, ArvAlbum *cen)
{
    *no = (ArvAlbum*)malloc(sizeof(ArvAlbum));

    (*no)->info_1 = a;
    (*no)->Nkeys = 1;
    (*no)->esq = esq;
    (*no)->cen = cen;
    (*no)->dir = NULL;
}

void adicionaNoAlbum(ArvAlbum **no, Album info, ArvAlbum *filho)
{
    if (strcmp(info.titulo, (*no)->info_1.titulo) > 0)
    {
        (*no)->info_2 = info;
        (*no)->dir = filho;
    }else{
        (*no)->info_2 = (*no)->info_1;
        (*no)->info_1 = info;
        (*no)->dir = (*no)->cen;
        (*no)->cen = filho;
    }
    (*no)->Nkeys = 2;
}

void quebraNoAlbum(ArvAlbum **no, Album info, Album *sobe, ArvAlbum **no_maior,
ArvAlbum *filho)
{
    if (strcmp(info.titulo, (*no)->info_1.titulo) < 0)
    {
        *sobe = (*no)->info_1;
        (*no)->info_1 = info;
        criaNoAlbum(no_maior, (*no)->info_2, (*no)->cen, (*no)->dir);
        (*no)->cen = filho;
    }else if (strcmp(info.titulo, (*no)->info_2.titulo) < 0)
    {
       *sobe = info;
       criaNoAlbum(no_maior, (*no)->info_2, filho, (*no)->dir);
    }else{
        *sobe = (*no)->info_2;
        criaNoAlbum(no_maior, info, (*no)->dir, filho);
    }
    (*no)->dir = NULL;
    (*no)->Nkeys = 1;
```

```
}

ArvAlbum *insereArvAlbum(ArvAlbum **pai, ArvAlbum **raiz, Album info, Album *sobe)
{
    ArvAlbum *no_maior = NULL;
    if (*raiz == NULL)
        criaNoAlbum(raiz, info, NULL, NULL);
    else{
        if (ehFolhaAlbum(*raiz) == 1)
        {
            if ((*raiz)->Nkeys == 1)
                adicionaNoAlbum(raiz, info, NULL);
            else{
                quebraNoAlbum(raiz, info, sobe, &no_maior, NULL);
                if (*pai == NULL)
                {
                    ArvAlbum *aux;
                    criaNoAlbum(&aux, *sobe, *raiz, no_maior);
                    *raiz = aux;
                    no_maior = NULL;
                }
            }
        }else{
            if (strcmp(info.titulo, (*raiz)->info_1.titulo) < 0)
                no_maior = insereArvAlbum(raiz, &(*raiz)->esq, info, sobe);
            else if(strcmp(info.titulo, (*raiz)->info_2.titulo) > 0 && (*raiz)-
>Nkeys == 2)
                no_maior = insereArvAlbum(raiz, &(*raiz)->dir, info, sobe);
            else
                no_maior = insereArvAlbum(raiz, &(*raiz)->cen, info, sobe);
            if (no_maior != NULL)
            {
                if ((*raiz)->Nkeys == 1)
                {
                    adicionaNoAlbum(raiz, *sobe, no_maior);
                    no_maior = NULL;
                }else{
                    Album sobe2;
                    ArvAlbum *no_maior_2;

                    quebraNoAlbum(raiz, *sobe, &sobe2, &no_maior_2, no_maior);
                    if (*pai == NULL)
                    {
                        ArvAlbum *aux;
                        criaNoAlbum(&aux, sobe2, *raiz, no_maior_2);
                        *raiz = aux;
                        no_maior = NULL;
                    }else{
                        *sobe = sobe2;
                        no_maior = no_maior_2;
                    }
```

```
                }

            }

        }
    }
    return no_maior;
}

void imprimirArvAlbum(ArvAlbum *raiz)
{
    if (raiz != NULL)
    {
        imprimirArvAlbum(raiz->esq);

        printf("Titulo: %s\nAno: %d\nQuantidade de musicas: %d\n", raiz-
>info_1.titulo, raiz->info_1.ano, raiz->info_1.quantidade_musica);

        if (raiz->Nkeys == 2)
        {
            printf("Titulo: %s\nAno: %d\nQuantidade de musica: %d\n", raiz-
>info_2.titulo, raiz->info_2.ano, raiz->info_2.quantidade_musica);
        }

        imprimirArvAlbum(raiz->cen);
        imprimirArvAlbum(raiz->dir);
    }
}


/*--------------------------------------------------------------------------------*/

void liberaNoAlbum(ArvAlbum **no)
{
    ArvAlbum *aux;

    aux = *no;
    *no = NULL;
    free(aux);
}

int removeEsqFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai)
{
    int flag = 0;
    (**raiz).info_1 = (**pai).info_1;

    if ((**pai).cen->Nkeys == 2)
    {
        (**pai).info_1 = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).cen->info_2;
```

```
        (**pai).cen->Nkeys = 1;
    }else if((**pai).Nkeys == 2){
        (**pai).info_1 = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).info_2;
        if ((**pai).dir->Nkeys == 2)
        {
            (**pai).info_2 = (**pai).dir->info_1;
            (**pai).dir->info_1 = (**pai).dir->info_2;
            (**pai).dir->Nkeys = 1;
        }
        else{
            adicionaNoAlbum(&(**pai).cen, (**pai).dir->info_1, NULL);
            liberaNoAlbum(&(**pai).dir);
            (**pai).Nkeys = 1;
        }

    }else{
        adicionaNoAlbum(pai, (**pai).cen->info_1, NULL);
        (**pai).Nkeys = 2;
        liberaNoAlbum(&(**pai).esq);
        liberaNoAlbum(&(**pai).cen);
        flag = 1;
    }

    return flag;
}

int removeCenFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai)
{
    int flag = 0;

    if ((**pai).esq->Nkeys == 2)
    {
        (**raiz).info_1 = (**pai).info_1;
        (**pai).info_1 = (**pai).esq->info_2;
        (**pai).esq->Nkeys = 1;
    }else if ((**pai).Nkeys == 2)
    {
        (**raiz).info_1 = (**pai).info_2;
        if ((**pai).dir->Nkeys == 2)
        {
            (**pai).info_2 = (**pai).dir->info_1;
            (**pai).dir->info_1 = (**pai).dir->info_2;
            (**pai).dir->Nkeys = 1;
        }else{
            adicionaNoAlbum(&(**pai).cen, (**pai).dir->info_1, NULL);
            liberaNoAlbum(&(**pai).dir);
            (**pai).Nkeys = 1;
        }
    }else{
        adicionaNoAlbum(pai, (**pai).esq->info_1, NULL);
```

```
        (**pai).Nkeys = 2;
        liberaNoAlbum(&(**pai).esq);
        liberaNoAlbum(&(**pai).dir);
        flag = 1;
    }

    return flag;
}

void removeDirFolhaAlbum(ArvAlbum **raiz, ArvAlbum **pai)
{
    (**raiz).info_1 = (**pai).info_2;
    if ((**pai).cen->Nkeys == 2)
    {
        (**pai).info_2 = (**pai).cen->info_2;
        (**pai).cen->Nkeys = 1;
    }else if ((**pai).esq->Nkeys == 2)
    {
        Album aux = (**pai).cen->info_1;
        (**pai).cen->info_1 = (**pai).info_1;
        (**pai).info_1 = (**pai).esq->info_2;
        (**pai).info_2 = aux;
        (**pai).esq->Nkeys = 1;
    }else{
        adicionaNoAlbum(&(**pai).cen, (**pai).info_2, NULL);
        (**pai).Nkeys = 1;
        liberaNoAlbum(&(**pai).dir);
    }
}

int removeFolhaAlbum(ArvAlbum **pai, ArvAlbum **raiz, char *nome)
{
    int balanceamento = 0;
    if ((*raiz)->Nkeys == 2)
    {
        if (strcmp(nome, (*raiz)->info_1.titulo) == 0)
            (*raiz)->info_1 = (*raiz)->info_2;
        (*raiz)->Nkeys = 1;
    }else if (*pai == NULL)
    {
        liberaNoAlbum(raiz);
    }else{
        if (*raiz == (**pai).esq)
            balanceamento = removeEsqFolhaAlbum(raiz, pai);
        else if (*raiz == (**pai).cen)
            balanceamento = removeCenFolhaAlbum(raiz, pai);
        else if (*raiz == (**pai).dir)
            removeDirFolhaAlbum(raiz, pai);
    }

    return balanceamento;
```

```
}

int removeNoFilhoFolhaAlbum(ArvAlbum **raiz, char *nome)
{
    int balanceamento = 0;
    if (strcmp(nome, (**raiz).info_1.titulo) == 0)
    {
        if ((**raiz).cen->Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).cen->info_2;
            (**raiz).Nkeys = 1;
        }else if ((**raiz).esq->Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).esq->info_2;
            (**raiz).esq->Nkeys = 1;
        }else if ((**raiz).Nkeys == 2)
        {
            (**raiz).info_1 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).info_2;
            if ((**raiz).dir->Nkeys == 2)
            {
                (**raiz).info_2 = (**raiz).dir->info_1;
                (**raiz).dir->info_1 = (**raiz).dir->info_2;
                (**raiz).dir->Nkeys = 1;
            }else{
                adicionaNoAlbum(&(**raiz).cen, (**raiz).dir->info_1, NULL);
                (**raiz).Nkeys = 1;
                liberaNoAlbum(&(**raiz).dir);
            }

        }else{
            ArvAlbum *aux;
            adicionaNoAlbum(&(**raiz).esq, (**raiz).cen->info_1, NULL);
            aux = (**raiz).esq;
            liberaNoAlbum(&(**raiz).cen);
            liberaNoAlbum(raiz);
            *raiz = aux;
            balanceamento = 1;
        }
    }else{
        if ((**raiz).dir->Nkeys == 2)
        {
            (**raiz).info_2 = (**raiz).dir->info_1;
            (**raiz).dir->info_1 = (**raiz).dir->info_2;
            (**raiz).dir->Nkeys = 1;
        }else if ((**raiz).cen->Nkeys == 2)
        {
            (**raiz).info_2 = (**raiz).cen->info_2;
            (**raiz).cen->Nkeys = 1;
        }else if ((**raiz).esq->Nkeys == 2)
```

```
        {
            (**raiz).info_2 = (**raiz).cen->info_1;
            (**raiz).cen->info_1 = (**raiz).info_1;
            (**raiz).info_1 = (**raiz).esq->info_2;
            (**raiz).esq->Nkeys = 1;
        }else{
            adicionaNoAlbum(&(**raiz).cen, (**raiz).dir->info_1, NULL);
            (**raiz).Nkeys = 1;
            liberaNoAlbum(&(**raiz).dir);
        }
    }

    return balanceamento;
}

int balanceiaArvAlbum(ArvAlbum **raiz, int desbalanceamento)
{
    Album sobe;
    ArvAlbum *no_maior;
    ArvAlbum *aux;

    if (desbalanceamento == -1)
    {
        aux = (*raiz)->cen->esq;
        (*raiz)->cen->esq = (*raiz)->esq;
        if ((*raiz)->cen->Nkeys == 1)
        {
            adicionaNoAlbum(&(*raiz)->cen, (*raiz)->info_1, aux);
            (*raiz)->esq = NULL;
            if ((*raiz)->Nkeys == 1)
            {
                aux = *raiz;
                *raiz = (*raiz)->cen;
                liberaNoAlbum(&aux);
            }else{
                (*raiz)->info_1 = (*raiz)->info_2;
                (*raiz)->esq = (*raiz)->cen;
                (*raiz)->cen = (*raiz)->dir;
                (*raiz)->dir = NULL;
                (*raiz)->Nkeys = 1;
            }

        }else{
            quebraNoAlbum(&(*raiz)->cen, (*raiz)->info_1, &sobe, &no_maior, aux);
            (*raiz)->info_1 = sobe;
            (*raiz)->esq = (*raiz)->cen;
            (*raiz)->cen = no_maior;
        }
    }else if (desbalanceamento == 0)
    {
        if ((*raiz)->esq->Nkeys == 1)
```

```
        {
            adicionaNoAlbum(&(*raiz)->esq, (*raiz)->info_1, (*raiz)->cen);
            (*raiz)->cen = NULL;
            if ((*raiz)->Nkeys == 1)
            {
                aux = *raiz;
                *raiz = (*raiz)->esq;
                liberaNoAlbum(&aux);
            }else{
                (*raiz)->info_1 = (*raiz)->info_2;
                (*raiz)->cen = (*raiz)->dir;
                (*raiz)->dir = NULL;
                (*raiz)->Nkeys = 1;
            }

        }else{
            quebraNoAlbum(&(*raiz)->esq, (*raiz)->info_1, &sobe, &no_maior,
(*raiz)->cen);
            (*raiz)->info_1 = sobe;
            (*raiz)->cen = no_maior;
        }
    }else{
        if ((*raiz)->cen->Nkeys == 1)
        {
            adicionaNoAlbum(&(*raiz)->cen, (*raiz)->info_2, (*raiz)->dir);
            (*raiz)->Nkeys = 1;
            (*raiz)->dir = NULL;
        }else{
            quebraNoAlbum(&(*raiz)->cen, (*raiz)->info_2, &sobe, &no_maior,
(*raiz)->dir);
            (*raiz)->info_2 = sobe;
            (*raiz)->dir = no_maior;
        }
    }

    return 0;
}

int removeMenorNoAlbum(ArvAlbum **pai_aux, ArvAlbum **no, Album *info_sobe)
{
    int balanceamento = 0;
    if ((*no)->esq != NULL)
        balanceamento = removeMenorNoAlbum(no, &(*no)->esq, info_sobe);
    else{
        *info_sobe = (*no)->info_1;
        balanceamento = removeFolhaAlbum(pai_aux, no, (*no)->info_1.titulo);
    }
    return balanceamento;
}

int removerAlbum(ArvAlbum **pai, ArvAlbum **raiz, char *nome)
```

```
{
    int balanceamento = 0;
    if (*raiz != NULL)
    {
        if ((strcmp(nome, (*raiz)->info_1.titulo) == 0) || ((*raiz)->Nkeys == 2 &&
strcmp(nome, (*raiz)->info_2.titulo) == 0))
        {

            if (ehFolhaAlbum(*raiz) == 1)
                balanceamento = removeFolhaAlbum(pai, raiz, nome);
            else if (((ehFolhaAlbum((*raiz)->esq) == 1) && ehFolhaAlbum((*raiz)-
>cen) == 1 && ehFolhaAlbum((*raiz)->dir) == 1) || ((ehFolhaAlbum((*raiz)->esq) ==
1) && ehFolhaAlbum((*raiz)->cen) == 1 && (*raiz)->Nkeys == 1 ))
                balanceamento = removeNoFilhoFolhaAlbum(raiz, nome);
            else{
                ArvAlbum *pai_aux = NULL;
                Album info_sobe;
                if (strcmp(nome, (*raiz)->info_1.titulo) == 0)
                {
                    balanceamento = removeMenorNoAlbum(&pai_aux, &(*raiz)->cen,
&info_sobe);
                    (*raiz)->info_1 = info_sobe;
                    if (balanceamento == 1)
                    {
                        balanceamento = balanceiaArvAlbum(raiz, 0);
                    }
                }else{
                    balanceamento = removeMenorNoAlbum(&pai_aux, &(*raiz)->dir,
&info_sobe);
                    (*raiz)->info_2 = info_sobe;
                    if (balanceamento == 1)
                    {
                        balanceamento = balanceiaArvAlbum(raiz, 1);
                    }
                }
            }
        }
        else if (strcmp(nome,(*raiz)->info_1.titulo) < 0)
            balanceamento = removerAlbum(raiz, &(*raiz)->esq, nome);
        else if (strcmp(nome,(*raiz)->info_2.titulo) > 0 && (*raiz)->Nkeys == 2)
            balanceamento = removerAlbum(raiz, &(*raiz)->dir, nome);
        else
            balanceamento = removerAlbum(raiz, &(*raiz)->cen, nome);
        if (balanceamento == 1 && *raiz != NULL && *pai != NULL)
        {
            if (*raiz == (**pai).esq)
                balanceamento = balanceiaArvAlbum(pai, -1);
            else if(*raiz == (*pai)->cen)
                balanceamento = balanceiaArvAlbum(pai, 0);
            else if(*raiz == (*pai)->dir)
                balanceamento = balanceiaArvAlbum(pai, 1);
```

```
        }

    }
    return balanceamento;
}

ArvAlbum *buscaArvAlbum(ArvAlbum **raiz, char *nome)
{
    ArvAlbum *no = NULL;
    if (*raiz != NULL)
    {
        if (strcmp(nome, (*raiz)->info_1.titulo) == 0)
            no = *raiz;
        else if(strcmp(nome, (*raiz)->info_2.titulo) == 0)
            no = *raiz;
        else
        {
            if (strcmp(nome, (*raiz)->info_1.titulo) < 0)
                no = buscaArvAlbum(&(*raiz)->esq, nome);
            else if (strcmp(nome, (*raiz)->info_2.titulo) < 0 || (*raiz)->Nkeys ==
1)
                no = buscaArvAlbum(&(*raiz)->cen, nome);
            else
                no = buscaArvAlbum(&(*raiz)->dir, nome);
        }
    }

    return no;
}
/*--------------------------------------------------------------------------------*/
Musica ler_musica()
{
    Musica musica;
    fflush(stdin);
    printf("Informe o nome da musica: ");
    scanf("%[^\n]s", musica.nome_musica);
    getchar();
    fflush(stdin);
    printf("Informe quantidade de minutos: ");
    scanf("%d", &musica.minutos);
    return musica;
}


Musica *criarLista()
{
    return NULL;
}

void inserirMusica(ArvAlbum *lista)
```

```
{
    Musica *m = (Musica *)malloc(sizeof(Musica));

    *m = ler_musica();
    m->prox = NULL;

    Musica *atual = lista->musica;
    Musica *anterior = NULL;

    while (atual != NULL && strcmp(m->nome_musica, atual->nome_musica) > 0)
    {
        anterior = atual;
        atual = atual->prox;
    }

    if (anterior == NULL)
    {
        m->prox = lista->musica;
        lista->musica = m;
    }
    else
    {
        anterior->prox = m;
        m->prox = atual;
    }
}

Musica *busca_musica(ArvAlbum *lista, char *nome)
{
    Musica *atual = lista->musica;

    while (atual != NULL && strcmp(atual->nome_musica, nome) != 0)
        atual = atual->prox;

    return atual;
}

void excluir_musica(ArvAlbum *lista, char *nome)
{
    Musica *anterior = NULL;
    Musica *atual = lista->musica;

    while (atual != NULL && strcmp(atual->nome_musica, nome) != 0)
    {
        anterior = atual;
        atual = atual->prox;
    }
    if (atual != NULL)
    {
        if (anterior == NULL)
            lista->musica = atual->prox;
```

```
        else
            anterior->prox = atual->prox;
        free(atual);
    }
}

void imprimir_todas_musicas(ArvAlbum *lista)
{
    Musica *m = lista->musica;
    while (m != NULL)
    {
        imprimir_uma_musica(m);
        m = m->prox;
    }
}

void imprimir_uma_musica(Musica* m)
{
    printf("Nome da musica: %s\nDuracao em minutos: %d min\n", m->nome_musica, m-
>minutos);
}

void imprimir_um_artista(Artista info)
{
    printf("Nome do artista: %s\nEstilo musical: %s\nQuantidade de albuns: %d\n",
info.nome_artista, info.estilo, info.num_albuns);
}

void imprimir_um_album(Album info)
{
    printf("Titulo: %s\nAno: %d\nQuantidade de musicas: %d\n", info.titulo,
info.ano, info.quantidade_musica);
}
/*--------------------------------------------------------------------------------*/

void menu()
{
    printf("==============MENU==============\n");
    printf(": 1 - Cadastro de Artista      :\n");
    printf(": 2 - Excluir Artista          :\n");
    printf(": 3 - Imprimir Artistas        :\n");
    printf(": 4 - Busca                    :\n");
    printf(": 5 - Cadastro de Album        :\n");
    printf(": 6 - Excluir Album            :\n");
    printf(": 7 - Imprimir Albuns          :\n");
    printf(": 8 - Cadastro de Musica       :\n");
    printf(": 9 - Excluir Musica           :\n");
    printf(": 10 - Imprimir Musica         :\n");
    printf(": 0 - Sair                     :\n");
    printf("================================\n");
}
```

```c
void menu_busca()
{
    system("cls");
    printf("==============MENU==============\n");
    printf(": 1 - Buscar Artista           :\n");
    printf(": 2 - Buscar Album             :\n");
    printf(": 3 - Buscar Musica            :\n");
    printf(": 0 - Sair                     :\n");
    printf("================================\n");
}



int main()
{
    Artista sobe;
    ArvAlbum *paiAlbum = criaArvAlbum();
    ArvArtista *pai = criaArvArtista();
    ArvArtista *raiz = criaArvArtista();


    int op = -1;

    while (op != 0)
    {
        menu();
        scanf("%d", &op);
        switch (op)
        {
            case 1:
            {
                insereArvArtista(&pai, &raiz, ler_artista(), &sobe);
                printf("Artista inserido.\n");
                break;
            }

            case 2:
            {
                fflush(stdin);
                int op_excluir;
                printf("Todos os albuns e musicas do artista serao excluidos
juntamente. Continuar? \n1 - Sim\n2 - Nao\n");
                scanf("%d", &op_excluir);

                if (op_excluir == 1)
                {
                    fflush(stdin);
                    char nome[50];
                    fflush(stdin);
                    printf("Informe o artista a ser excluido: ");
```

```
                    scanf("%[^\n]", nome);
                    ArvArtista *busca = buscaArvArtista(&raiz, nome);
                    if (busca != NULL)
                    {
                        removerArtista(&pai, &raiz, nome);
                        printf("Artista excluido com sucesso!!\n");
                    }
                    else
                        printf("Artista nao encontrado!!\n");
                }else if(op_excluir == 2)
                    printf("Operacao cancelada!!\n");

                break;
            }

            case 3:
                imprimirArvArtista(raiz);
                break;
            case 4:
            {
                int op_busca;
                menu_busca();
                scanf("%d", &op_busca);
                if (op_busca == 1)
                {
                    char nome[50];
                    fflush(stdin);
                    printf("Informe o artista a ser buscado: ");
                    scanf("%[^\n]", nome);
                    fflush(stdin);
                    ArvArtista *busca = buscaArvArtista(&raiz, nome);
                    if (busca != NULL)
                        imprimir_um_artista(busca->info_1);
                    else
                        printf("Artista nao encontrado!!\n");
                }
                else if(op_busca == 2)
                {
                    char nome_artista[50];
                    fflush(stdin);
                    printf("Informe o artista que tenha o album a ser buscado: ");
                    scanf("%[^\n]", nome_artista);
                    fflush(stdin);
                    ArvArtista *busca = buscaArvArtista(&raiz, nome_artista);
                    if (busca != NULL)
                    {
                        char nome[50];
                        fflush(stdin);
                        printf("Informe o artista que tenha o album a ser buscado:
");
                        scanf("%[^\n]", nome);
```

```
                        fflush(stdin);
                        ArvAlbum *busca_album = buscaArvAlbum(&raiz->arvAlbum,
nome);
                        if (busca_album != NULL)
                            imprimir_um_album(busca_album->info_1);
                        else
                            printf("Album nao encontrado!!\n");
                    }
                    else
                        printf("Artista nao encontrado!!\n");


                }
                else if (op_busca == 3)
                {

                    char nome_artista[50];
                    fflush(stdin);
                    printf("Informe o artista que tenha o album a ser buscado: ");
                    scanf("%[^\n]", nome_artista);
                    fflush(stdin);
                    ArvArtista *busca = buscaArvArtista(&raiz, nome_artista);
                    if (busca != NULL)
                    {
                        char nome[50];
                        fflush(stdin);
                        printf("Informe o artista que tenha o album a ser buscado:
");
                        scanf("%[^\n]", nome);
                        fflush(stdin);
                        ArvAlbum *busca_album = buscaArvAlbum(&raiz->arvAlbum,
nome);
                        if (busca_album != NULL)
                        {
                            char nome_musica[50];
                            printf("Informe o nome da musica: ");
                            scanf("%[^\n]", nome_musica);
                            Musica *mus = busca_musica(busca_album, nome_musica);
                            if (mus != NULL)
                                imprimir_uma_musica(mus);
                            else
                                printf("Musica nao encontrada!!\n");

                        }
                        else
                            printf("Album nao encontrado!!\n");

                    }
                    else
                        printf("Artista nao encontrado!!\n");
                }
```

```
                break;
            }

            case 5:
            {

                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja cadastrar um album: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvArtista(&raiz, nome);
                if (busca != NULL)
                {


                    ArvAlbum *raizAlbum = busca->arvAlbum;
                    Album sobeAlbum;
                    insereArvAlbum(&paiAlbum, &raizAlbum, ler_album(), &sobeAlbum);
                    busca->info_1.num_albuns++;
                    printf("Album inserido.\n");
                }
                else
                    printf("Artista nao encontrado!!\n");

                break;
            }

            case 6:
            {
                fflush(stdin);
                int op_excluir;
                printf("Todos as musicas do album serao excluidos juntamente.
Continuar? \n1 - Sim\n2 - Nao\n");
                scanf("%d", &op_excluir);
                if (op_excluir == 1)
                {

                    char nome[50];
                    fflush(stdin);
                    printf("Informe o artista que deseja excluir o album: ");
                    scanf("%[^\n]", nome);
                    fflush(stdin);
                    ArvArtista *busca = buscaArvArtista(&raiz, nome);
                    if (busca != NULL)
                    {
                        char titulo[50];
                        fflush(stdin);
                        printf("Informe o album a ser excluido: ");
                        scanf("%[^\n]", titulo);
                        fflush(stdin);
```

```
                        ArvAlbum *busca_album = buscaArvAlbum(&raiz->arvAlbum,
titulo);
                        if (busca_album != NULL)
                        {
                            removerAlbum(&paiAlbum, &busca->arvAlbum, titulo);
                            busca->info_1.num_albuns--;
                            printf("Album excluido com sucesso!!\n");
                        }
                        else
                            printf("Album nao encontrado!!\n");
                    }
                    else
                        printf("Artista nao encontrado!!\n");

                }else
                    printf("Operacao cancelada!!\n");

                break;
            }

            case 7:
            {

                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja mostrar os albuns: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvArtista(&raiz, nome);
                if (busca != NULL)
                    imprimirArvAlbum(busca->arvAlbum);
                else
                    printf("Artista nao encontrado!!\n");

                break;
            }
            case 8:
            {
                fflush(stdin);

                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja cadastrar a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvArtista(&raiz, nome);
                if (busca != NULL)
                {
                    char nome_album[50];
                    fflush(stdin);
                    printf("Informe o album que deseja cadastrar musica: ");
```

```
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvAlbum(&busca->arvAlbum,
nome_album);
                    if (busca_album != NULL)
                    {
                        inserirMusica(busca_album);
                        printf("Musica inserida com sucesso!!\n");
                    }
                    else
                        printf("Album nao encontrado!!\n");

                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            case 9:
            {

                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja excluir a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvArtista(&raiz, nome);
                if (busca != NULL)
                {
                    char nome_album[50];
                    fflush(stdin);
                    printf("Informe o album que deseja excluir a musica: ");
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvAlbum(&busca->arvAlbum,
nome_album);
                    if (busca_album != NULL)
                    {
                        char nome_musica[50];

                        fflush(stdin);
                        printf("Informe a musica que deseja excluir: ");
                        scanf("%[^\n]", nome_musica);
                        Musica *buscaMusica = busca_musica(busca_album,
nome_musica);
                        if (buscaMusica != NULL)
                        {
                            excluir_musica(busca_album, nome_musica);
                            busca_album->info_1.quantidade_musica--;
                            printf("Musica deletada com sucesso!!\n");
                        }
                        else
                            printf("Musica nao encontrada!!\n");
                    }
```

```
                    else
                        printf("Album nao encontrado!!\n");

                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            case 10:
            {

                char nome[50];
                fflush(stdin);
                printf("Informe o artista que deseja mostrar a musica: ");
                scanf("%[^\n]", nome);
                fflush(stdin);
                ArvArtista *busca = buscaArvArtista(&raiz, nome);
                if (busca != NULL)
                {
                    char nome_album[50];
                    fflush(stdin);
                    printf("Informe o album que deseja mostrar as musicas: ");
                    scanf("%[^\n]", nome_album);
                    ArvAlbum *busca_album  = buscaArvAlbum(&busca->arvAlbum,
nome_album);
                    if (busca_album != NULL)
                        imprimir_todas_musicas(busca_album);
                    else
                        printf("Album nao encontrado!!\n");
                }
                else
                    printf("Artista nao encontrado!!\n");
                break;
            }
            default:
                break;
        }
    }


    return 0;
}
```

Questão 2.1:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
```

```c
typedef struct artista {
    char nome[50];
} Artista;

typedef struct arvartista {
    Artista artistas[2];
    int Nkeys;
    struct arvartista *esq, *cen, *dir;
} ArvArtista;

ArvArtista *criaNo(Artista artista1, Artista artista2) {
    ArvArtista *novoNo = (ArvArtista *)malloc(sizeof(ArvArtista));
    novoNo->artistas[0] = artista1;
    novoNo->artistas[1] = artista2;
    novoNo->Nkeys = (artista2.nome[0] != '\0') ? 2 : 1;
    novoNo->esq = novoNo->cen = novoNo->dir = NULL;
    return novoNo;
}

ArvArtista *insereNo(ArvArtista *tree, ArvArtista *novoNo) {
    if (tree == NULL) {
        return novoNo;
    }

    if (strcmp(novoNo->artistas[0].nome, tree->artistas[0].nome) < 0) {
        tree->esq = insereNo(tree->esq, novoNo);
    } else if (strcmp(novoNo->artistas[0].nome, tree->artistas[0].nome) > 0) {
        tree->dir = insereNo(tree->dir, novoNo);
    } else if (novoNo->artistas[1].nome[0] != '\0') {
        tree->cen = insereNo(tree->cen, novoNo);
    }

    return tree;
}

ArvArtista *buscaTestes(ArvArtista *tree, char *codigo, int n) {
    if (tree != NULL) {
        if (strcmp(codigo, tree->artistas[0].nome) == 0) {
            printf("%d-left ", n);
            return tree;
        } else if (tree->Nkeys == 2 && strcmp(codigo, tree->artistas[1].nome) == 0)
{
            printf("%d-center ", n);
            return tree;
        } else {
            if (strcmp(codigo, tree->artistas[0].nome) < 0) {
                printf("%d-left ", n);
                return buscaTestes(tree->esq, codigo, n + 1);
            } else if (tree->Nkeys == 1 || strcmp(codigo, tree->artistas[1].nome) <
0) {
                printf("%d-center ", n);
```

```
                return buscaTestes(tree->cen, codigo, n + 1);
            } else {
                printf("%d-right ", n);
                return buscaTestes(tree->dir, codigo, n + 1);
            }
        }
    }
    return NULL;
}

void liberaArvore(ArvArtista *tree) {
    if (tree != NULL) {
        liberaArvore(tree->esq);
        liberaArvore(tree->cen);
        liberaArvore(tree->dir);
        free(tree);
    }
}

int main() {
    ArvArtista *tree = NULL;

    char *artistas[30] = {
        "Ana", "Beatriz", "Carlos", "Daniel", "Eduardo", "Fernanda", "Gabriel",
"Helena", "Igor",
        "Julia", "Kleber", "Laura", "Marcos", "Natalia", "Otavio", "Patricia",
"Quirino", "Rafael",
        "Sabrina", "Tiago", "Ursula", "Vitor", "Wagner", "Ximena", "Yasmin",
"Zacarias",
        "Alice", "Bruno", "Cecilia", "Diego"};

    double total_time = 0;

    for (int i = 0; i < 30; i++) {
        Artista artista1, artista2;
        strncpy(artista1.nome, artistas[i], sizeof(artista1.nome) - 1);
        artista1.nome[sizeof(artista1.nome) - 1] = '\0';
        artista2.nome[0] = '\0';

        ArvArtista *novoNo = criaNo(artista1, artista2);
        tree = insereNo(tree, novoNo);

        int n = 0;
        printf("Caminho percorrido para encontrar '%s': ", artistas[i]);

        clock_t start_time = clock();
        ArvArtista *resultado = buscaTestes(tree, artistas[i], n);
        clock_t end_time = clock();

        double elapsed_time = ((double)(end_time - start_time) / CLOCKS_PER_SEC) *
1000000000;
```

```
        total_time += elapsed_time;

        printf("\n");

        if (resultado != NULL) {
            printf("'%s' encontrado!\n", artistas[i]);
        } else {
            printf("'%s' nao encontrado.\n", artistas[i]);
        }

        printf("Tempo de execucao: %.2f nanossegundos\n", elapsed_time);
    }

    printf("Media do tempo total de execucao: %.2f nanossegundos\n",
total_time/30);

    liberaArvore(tree);

    return 0;
}
```

Questão 3:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int Tam_bloco;

struct Bloco
{
    int Bloco_I, Bloco_F;
    char status;
};

struct Arv45
{
    struct Bloco info1, info2, info3, info4;
    int Ninfos;
    struct Arv45 *Esq, *Cen_Esq, *Cen, *Cen_Dir, *Dir;
};

typedef struct Bloco Bloco;
typedef struct Arv45 Arv45;

void Cadastra(Arv45 **raiz);
Arv45 *insere(Arv45 **raiz, Bloco valor, Bloco *promove, Arv45 **pai);
Arv45 *criaNo(Bloco valor, Arv45 *filho_esq, Arv45 *filho_cen_esq);
int ehFolha(Arv45 *raiz);
Arv45 *adicionaChave(Arv45 *raiz, Bloco valor, Arv45 *ptr);
Arv45 *quebraNo(Arv45 *raiz, Bloco valor, Bloco *sobe, Arv45 *ptr);
```

```c
void imprimir(Arv45 *raiz);
void exibirInfo(Bloco *valor);
void AlocaEspacos(Arv45 **raiz, int Qtd);
Bloco ProcuraEspaco(Arv45 **raiz, int Qtd, Arv45 **pai, Arv45 **RAIZAvr, int *ptr);
Arv45 *busca45(Arv45 **raiz, int info);
int remover(Arv45 **raiz, Arv45 **pai, int valor);
int balancear(Arv45 **raiz, Arv45 **pai, int removeu);
int removeInfo2Folha(Arv45 *raiz);
int removeInfo2ComDirFolha(Arv45 *raiz, Arv45 *pai);
Arv45 *obterMenorNo(Arv45 *raiz);
int numeroInfosArv(Arv45 *raiz);
Arv45 *obterMaiorNo(Arv45 *raiz);
int removeInfo2NaoFolha(Arv45 *raiz);
int removeInfo1Folha(Arv45 *raiz, Arv45 *pai);
int removeInfo1NaoFolha(Arv45 *raiz);
void liberarMemoriaArvore(Arv45 *raiz);

int main(){
    Arv45 *raiz = NULL;
    int op, Qtd, aux;

    Cadastra(&raiz);
    printf("\nCadastro Finalizado!\n\n");
    do
    {
        printf("\n---- Menu ----\n1 - Alocar Nos\n2 - Remover blocos\n3 -
Imprimir\n0 - Sair\n");
        scanf("%d", &op);
        printf("\n");
        switch (op)
        {
        case 0:
            printf("Saindo!!!\n\n");
            break;
        case 1:
            printf("Alocar Nos\n");
            printf("\nDigite a quantidade de blocos que voce deseja alocar: ");
            scanf("%d", &Qtd);

            while (Qtd <= 0)
            {
                printf("\nDigite uma quantidade de blocos valida, maior que zero:
");
                scanf("%d", &Qtd);
            }

            AlocaEspacos(&raiz, Qtd);
            break;
        case 3:
            // system("cls");
            printf("Imprimindo blocos(Nos)\n");
```

```
            imprimir(raiz);
            break;
        default:
            printf("Digite uma opcao valida!\n");
            break;
        }
    } while (op != 0);
    liberarMemoriaArvore(raiz);
    return 0;
}

void Cadastra(Arv45 **raiz)
{
    Arv45 *pai = NULL;
    Bloco info, promove;
    int i = 0;

    system("cls");
    printf("\n\t-------- Cadastro da memoria --------\n");
    printf("\nDigite a quantidade de blocos logicos(Memoria em Mbyte): ");
    scanf("%d", &Tam_bloco);

    while (Tam_bloco <= 0)
    {
        printf("\nDigite uma quantidade de blocos valida, maior que zero: ");
        scanf("%d", &Tam_bloco);
    }

    printf("\nDigite o estado do primeiro bloco(o para ocupado ou 1 para livre):
");
    scanf(" %c", &info.status);

    while (info.status != 'o' && info.status != '1')
    {
        printf("\nDigite um estado valido para o primeiro bloco(o para ocupado ou 1
para livre): ");
        scanf(" %c", &info.status);
    }

    info.Bloco_I = 0;
    info.Bloco_F = 0;

    do
    {
        printf("\nInicio do bloco: %d\n", info.Bloco_I);
        printf("Digite o endereco final para o bloco: ");
        scanf("%d", &info.Bloco_F);

        while (info.Bloco_F >= Tam_bloco || info.Bloco_F < info.Bloco_I)
        {
```

```
            printf("Digite um valor valido para o fim do bloco(Fim do bloco: %d):
", Tam_bloco);
            scanf("%d", &info.Bloco_F);
        }

        // vet[i] = info.Bloco_F;
        // i++;

        insere(raiz, info, &promove, &pai);

        if (info.status == 'l')
        {
            info.status = 'o';
        }
        else
        {
            info.status = 'l';
        }

        info.Bloco_I = info.Bloco_F + 1;

    } while (info.Bloco_I != Tam_bloco);
}

Arv45 *insere(Arv45 **raiz, Bloco valor, Bloco *promove, Arv45 **pai){
    Arv45 *MaiorNo;
    MaiorNo = NULL;
    Bloco promove1;

    if (*raiz == NULL){
        *raiz = criaNo(valor, NULL, NULL);
    }else{
        if (ehFolha(*raiz)){ // e folha
            if ((*raiz)->Ninfos <
4){                                             // e folha, tem apenas uma
informacao
                *raiz = adicionaChave(*raiz, valor, NULL); // adiciona a informacao
ao no
                MaiorNo = NULL;
            }
            else
            { // e folha, tem duas informacoes
                MaiorNo = quebraNo(*raiz, valor, promove, NULL);
                if (*pai == NULL){ // cria uma nova raiz para a arvore
                    *raiz = criaNo(*promove, *raiz, MaiorNo);
                    MaiorNo = NULL; // O MaiorNo no recebe null, pois ja foi
adicionado na arvore, se isso nao acontecer ela vai inserir o elemento repetidas
vezes na arvore
                }
            }
        }else{
```

```
            if(valor.Bloco_F < (*raiz)->info1.Bloco_F)
                MaiorNo = insere(&((*raiz)->Esq), valor, promove, raiz);
            else if((*raiz)->Ninfos == 1 || ((*raiz)->Ninfos <= 4 && valor.Bloco_F
< (*raiz)->info2.Bloco_F))
                MaiorNo = insere(&((*raiz)->Cen_Esq), valor, promove, raiz);
            else if((*raiz)->Ninfos == 2 || ((*raiz)->Ninfos <= 4 && valor.Bloco_F
< (*raiz)->info3.Bloco_F))
                MaiorNo = insere(&((*raiz)->Cen), valor, promove, raiz);
            else if((*raiz)->Ninfos == 3 || ((*raiz)->Ninfos <= 4 && valor.Bloco_F
< (*raiz)->info4.Bloco_F))
                MaiorNo = insere(&((*raiz)->Cen_Dir), valor, promove, raiz);
            else
                MaiorNo = insere(&((*raiz)->Dir), valor, promove, raiz);

            if (MaiorNo){
                if((*raiz)->Ninfos < 4){
                    *raiz = adicionaChave(*raiz, *promove, MaiorNo);
                    MaiorNo = NULL;
                }else{
                    MaiorNo = quebraNo(*raiz, *promove, &promove1, MaiorNo);
                    // -----
                    *promove = promove1;
                    if(pai == NULL){
                        *raiz = criaNo(promove1, *raiz, MaiorNo);
                        MaiorNo = NULL;
                    }
                }
            }
        }
    }
    return MaiorNo;
}

Arv45 *criaNo(Bloco valor, Arv45 *filho_esq, Arv45 *filho_cen_esq){
    Arv45 *no;
    no = (Arv45 *)malloc(sizeof(Arv45));
    if (no != NULL){
        no->info1 = valor;
        no->Ninfos = 1;
        no->Esq = filho_esq;
        no->Cen_Esq = filho_cen_esq;
        no->Cen = NULL;
        no->Cen_Dir = NULL;
        no->Dir = NULL;

    }
    return no;
}

int ehFolha(Arv45 *raiz)
{ // verifica se o no e folha
```

```
    int ehfolha;
    if (raiz->Esq == NULL)
    { // se a esquerda for nula, significa que o no e folha, pois como nao tem esq, nao tera centro, nem direita
        ehfolha = 1;
    }
    else
    {
        ehfolha = 0;
    }
    return ehfolha; // retorna 1 se for folha e 0 se nao for
}

Arv45 *adicionaChave(Arv45 *raiz, Bloco valor, Arv45 *ptr){
    if ((raiz)->Ninfos == 1){
        if(valor.Bloco_F > raiz->info1.Bloco_F){
            raiz->info2 = valor;
            raiz->Cen = ptr;
        }else{
            raiz->info2 = raiz->info1;
            raiz->info1 = valor;

            raiz->Cen = raiz->Cen_Esq;
            raiz->Cen_Esq = ptr;
        }
        raiz->Ninfos = 2;
    }else if((raiz)->Ninfos ==2){
        if (valor.Bloco_F > raiz->info2.Bloco_F){
            raiz->info3 = valor;
            raiz->Cen_Dir = ptr;
        }else if(valor.Bloco_F > raiz->info1.Bloco_F && valor.Bloco_F < raiz->info2.Bloco_F){
            raiz->info3 = raiz->info2;
            raiz->info2 = valor;

            raiz->Cen_Dir = raiz->Cen;
            raiz->Cen = ptr;

        }else{
            raiz->info3 = raiz->info2;
            raiz->info2 = raiz->info1;
            raiz->info1 = valor;

            raiz->Cen_Dir = raiz->Cen;
            raiz->Cen = raiz->Cen_Esq;
            raiz->Cen_Esq = ptr;
        }
        raiz->Ninfos = 3;
    }else{
        if(valor.Bloco_F > raiz->info3.Bloco_F){
            raiz->info4 = valor;
```

```
            raiz->Dir = ptr;

        }else if(valor.Bloco_F > raiz->info2.Bloco_F && valor.Bloco_F < raiz-
>info3.Bloco_F){
            raiz->info4 = raiz->info3;
            raiz->info3 = valor;

            raiz->Dir = raiz->Cen_Dir;
            raiz->Cen_Dir = ptr;

        }else if (valor.Bloco_F > raiz->info1.Bloco_F && valor.Bloco_F < raiz-
>info2.Bloco_F){
            raiz->info4 = raiz->info3;
            raiz->info3 = raiz->info2;
            raiz->info2 = valor;

            raiz->Dir = raiz->Cen_Dir;
            raiz->Cen_Dir = raiz->Cen;
            raiz->Cen = ptr;

        }else{
            raiz->info4 = raiz->info3;
            raiz->info3 = raiz->info2;
            raiz->info2 = raiz->info1;
            raiz->info1 = valor;

            raiz->Dir = raiz->Cen_Dir;
            raiz->Cen_Dir = raiz->Cen;
            raiz->Cen = raiz->Cen_Esq;
            raiz->Cen_Esq = ptr;
        }
        raiz->Ninfos = 4;
    }
    return raiz;
}

Arv45 *quebraNo(Arv45 *raiz, Bloco valor, Bloco *sobe, Arv45 *ptr){
    Arv45 *maiorNo;
    if(valor.Bloco_F > raiz->info4.Bloco_F){
        *sobe = raiz->info3;

        maiorNo = criaNo(raiz->info4, raiz->Cen_Dir, raiz->Dir);

        maiorNo->info2 = valor;

        maiorNo->Cen = ptr;

        maiorNo->Ninfos = 2;
    }else if(valor.Bloco_F > raiz->info3.Bloco_F){
        *sobe = raiz->info3;
```

```
        maiorNo = criaNo(valor, raiz->Cen_Dir, ptr);

        maiorNo->info2 = raiz->info4;

        maiorNo->Cen = raiz->Dir;

        maiorNo->Ninfos = 2;
    }else if(valor.Bloco_F > raiz->info2.Bloco_F){
        *sobe = valor;

        maiorNo = criaNo(raiz->info3, raiz->Cen, raiz->Cen_Dir);

        maiorNo->info2 = raiz->info4;

        maiorNo->Cen = raiz->Dir;

        maiorNo->Ninfos = 2;
    }else if(valor.Bloco_F > raiz->info1.Bloco_F){
        *sobe = raiz->info2;

        maiorNo = criaNo(raiz->info3, raiz->Cen, raiz->Cen_Dir);

        maiorNo->info2 = raiz->info4;

        maiorNo->Cen = raiz->Dir;

        maiorNo->Ninfos = 2;

        raiz->info2 = valor;
        raiz->Cen = ptr;
    }else{
        *sobe = raiz->info2;

        maiorNo = criaNo(raiz->info3, raiz->Cen, raiz->Cen_Dir);

        maiorNo->info2 = raiz->info4;

        maiorNo->Cen = raiz->Dir;

        maiorNo->Ninfos = 2;

        raiz->info2 = raiz->info1;
        raiz->Cen = raiz->Cen_Esq;

        raiz->info1 = valor;
        raiz->Cen_Esq = ptr;
    }
    raiz->Ninfos = 2;
    raiz->Cen_Dir = NULL;
    raiz->Dir = NULL;
```

```
    return maiorNo;
}
void exibirInfo(Bloco *valor){
        // printf("cod: %d, tam: %d, qtd: %d, linha: %d, tipo: %s, marca: %s,
preco: %.2f\n", info->cod, info->tam, info->qtd, info->linha, info->tipo, info-
>marca, info->preco);
    printf("-Inicio: %d\n-Final: %d\n-Status: %c\n", valor->Bloco_I, valor-
>Bloco_F, valor->status);
}

void imprimir(Arv45 *raiz) {
    if (raiz != NULL) {
        imprimir(raiz->Esq);
        exibirInfo(&raiz->info1);
        if (raiz->Ninfos == 2 || raiz->Ninfos > 2)
            exibirInfo(&raiz->info2);
        if (raiz->Ninfos == 3 || raiz->Ninfos > 3)
            exibirInfo(&raiz->info3);
        if (raiz->Ninfos == 4)
            exibirInfo(&raiz->info4);

        imprimir(raiz->Cen_Esq);
        imprimir(raiz->Cen);
        imprimir(raiz->Cen_Dir);
        imprimir(raiz->Dir);
    }
}

void AlocaEspacos(Arv45 **raiz, int Qtd)
{
    Arv45 *Pai = NULL;
    Bloco aux;
    int ptr = 0, N;
    aux = ProcuraEspaco(&(*raiz), Qtd, &Pai, &(*raiz), &ptr);
    N = aux.Bloco_I;
    if (aux.Bloco_I != -1)
    {
        //remover(&(*raiz), &Pai, N);
    }
    else
    {
        printf("\nNao e possivel alocar essa quantidade de blocos, nao ha espaco
suficiente em nenhum bloco!\n");
    }
}

Bloco ProcuraEspaco(Arv45 **raiz, int Qtd, Arv45 **pai, Arv45 **RAIZAvr, int *ptr){
    Arv45 *aux1 = NULL, *aux2 = NULL;
    Bloco aux3;
    aux3.Bloco_I = -1;
    int espaco, aux = 0;
```

```
    if (*raiz != NULL){
        espaco = ((*raiz)->info1.Bloco_F - (*raiz)->info1.Bloco_I) + 1;
        if (espaco >= Qtd && (*raiz)->info1.status == 'l'){
            printf("\n---Bloco que foi Ocupado---\n");
            printf("  -Inicio: %d\n  -Final: %d\n  -Status: %c\n", (*raiz)-
>info1.Bloco_I, (*raiz)->info1.Bloco_F, (*raiz)->info1.status);
            aux = 1;
            if (espaco == Qtd){
                if (ehFolha(*raiz)){
                    if (*pai == NULL){
                        if ((*raiz)->Ninfos == 4){
                            (*raiz)->info1.Bloco_F = (*raiz)->info2.Bloco_F;
                            (*raiz)->info1.status = 'o';

                            (*raiz)->info2.Bloco_I = (*raiz)->info3.Bloco_I;
                            (*raiz)->info2.Bloco_F = (*raiz)->info3.Bloco_F;
                            (*raiz)->info2.status = (*raiz)->info3.status;

                            (*raiz)->info3.Bloco_I = (*raiz)->info4.Bloco_I;
                            (*raiz)->info3.Bloco_F = (*raiz)->info4.Bloco_F;
                            (*raiz)->info3.status = (*raiz)->info4.status;

                            (*raiz)->Ninfos = 3;
                            aux3.Bloco_I = -2;

                        }else if((*raiz)->Ninfos == 3){
                            (*raiz)->info1.Bloco_F = (*raiz)->info2.Bloco_F;
                            (*raiz)->info1.status = 'o';

                            (*raiz)->info2.Bloco_I = (*raiz)->info3.Bloco_I;
                            (*raiz)->info2.Bloco_F = (*raiz)->info3.Bloco_F;
                            (*raiz)->info2.status = (*raiz)->info3.status;

                            (*raiz)->Ninfos = 2;
                            aux3.Bloco_I = -2;

                        }else if ((*raiz)->Ninfos == 2){
                            (*raiz)->info1.Bloco_F = (*raiz)->info2.Bloco_F;
                            (*raiz)->info1.status = 'o';
                            (*raiz)->Ninfos = 1;
                            aux3.Bloco_I = -2;
                        }
                        else
                        {
                            (*raiz)->info1.status = 'o';
                            aux3.Bloco_I = -2;
                        }

                    }else{
                        if ((*raiz)->info1.Bloco_I == 0){
```

```
                        aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_F + 1);
                        if (aux1->Ninfos == 4){
                            if (aux1->info4.Bloco_I == (*raiz)->info3.Bloco_F +
1){

                                (*raiz)->info3.Bloco_F = aux1->info4.Bloco_F;
                                (*raiz)->info3.status = 'o';
                                aux3 = aux1->info2;

                                aux3 = aux1->info4;
                            }else if(aux1->info3.Bloco_I == (*raiz)-
>info2.Bloco_F + 1){

                                (*raiz)->info2.Bloco_F = aux1->info3.Bloco_F;
                                (*raiz)->info2.status = 'o';
                                aux3 = aux1->info2;

                                aux3 = aux1->info3;
                            }else if(aux1->info2.Bloco_I == (*raiz)-
>info1.Bloco_F + 1){

                                (*raiz)->info1.Bloco_F = aux1->info2.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info2;
                            }else{
                                (*raiz)->info1.Bloco_F = aux1->info1.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info1;
                            }
                        }else if(aux1->Ninfos == 3){
                            if (aux1->info3.Bloco_I == (*raiz)->info2.Bloco_F +
1){

                                (*raiz)->info2.Bloco_F = aux1->info3.Bloco_F;
                                (*raiz)->info2.status = 'o';

                                aux3 = aux1->info3;
                            }else if(aux1->info2.Bloco_I == (*raiz)-
>info1.Bloco_F + 1){

                                (*raiz)->info1.Bloco_F = aux1->info2.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info2;
                            }else{
                                (*raiz)->info1.Bloco_F = aux1->info1.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info1;
                            }
                        }else if(aux1->Ninfos == 2){
                            if (aux1->info2.Bloco_I == (*raiz)->info1.Bloco_F +
1)

                            {
                                (*raiz)->info1.Bloco_F = aux1->info2.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info2;
                            }else{
```

```
                                (*raiz)->info1.Bloco_F = aux1->info1.Bloco_F;
                                (*raiz)->info1.status = 'o';
                                aux3 = aux1->info1;
                            }
                        }else{
                            (*raiz)->info1.Bloco_F = aux1->info1.Bloco_F;
                            (*raiz)->info1.status = 'o';
                            aux3 = aux1->info1;
                        }
                    }else if ((*raiz)->info1.Bloco_F + 1 == Tam_bloco){
                        aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_I - 1);
                        if (aux1->Ninfos == 4){
                            if(aux1->info4.Bloco_F == (*raiz)->info3.Bloco_I -
1){

                                aux1->info4.Bloco_F = (*raiz)->info3.Bloco_F;
                                aux1->info4.status = 'o';
                                aux3 = (*raiz)->info3;
                            }else if(aux1->info3.Bloco_F == (*raiz)-
>info2.Bloco_I - 1){

                                aux1->info3.Bloco_F = (*raiz)->info2.Bloco_F;
                                aux1->info3.status = 'o';
                                aux3 = (*raiz)->info2;
                            }else if(aux1->info2.Bloco_F == (*raiz)-
>info1.Bloco_I - 1){

                                aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info2.status = 'o';
                                aux3 = (*raiz)->info1;
                            }else{
                                aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info1.status = 'o';
                                aux3 = (*raiz)->info1;
                            }
                        }else if(aux1->Ninfos ==3){
                            if(aux1->info4.Bloco_F == (*raiz)->info3.Bloco_I -
1){

                                aux1->info3.Bloco_F = (*raiz)->info2.Bloco_F;
                                aux1->info3.status = 'o';
                                aux3 = (*raiz)->info2;
                            }else if(aux1->info2.Bloco_F == (*raiz)-
>info1.Bloco_I - 1){

                                aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info2.status = 'o';
                                aux3 = (*raiz)->info1;
                            }else{
                                aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info1.status = 'o';
                                aux3 = (*raiz)->info1;
                            }
                        }else if (aux1->Ninfos == 2){
                            if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_I -
1)
```

```
                                {
                                    aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info2.status = 'o';
                                    aux3 = (*raiz)->info1;
                                }
                                else
                                {
                                    aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info1.status = 'o';
                                    aux3 = (*raiz)->info1;
                                }
                            }
                            else{
                                aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info1.status = 'o';
                                aux3 = (*raiz)->info1;
                            }
                        }else{
                            aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_I - 1);
                            if (aux1->Ninfos == 4){
                                if(aux1->info4.Bloco_F == (*raiz)->info3.Bloco_I -
1){

                                    aux1->info4.Bloco_F = (*raiz)->info3.Bloco_F;
                                    aux1->info4.status = 'o';
                                }else if(aux1->info3.Bloco_F == (*raiz)-
>info2.Bloco_I - 1){

                                    aux1->info3.Bloco_F = (*raiz)->info2.Bloco_F;
                                    aux1->info3.status = 'o';
                                }else if(aux1->info2.Bloco_F == (*raiz)-
>info1.Bloco_I - 1){

                                    aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info2.status = 'o';
                                }else
                                {
                                    aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info1.status = 'o';
                                }
                            }else if(aux1->Ninfos == 3){
                                if(aux1->info3.Bloco_F == (*raiz)->info2.Bloco_I -
1){

                                    aux1->info3.Bloco_F = (*raiz)->info2.Bloco_F;
                                    aux1->info3.status = 'o';
                                }else if(aux1->info2.Bloco_F == (*raiz)-
>info1.Bloco_I - 1){

                                    aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info2.status = 'o';
                                }else
                                {
                                    aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                    aux1->info1.status = 'o';
                                }
```

```
                        }else if (aux1->Ninfos == 2){
                            if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_I -
1){
                                aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info2.status = 'o';
                            }
                            else
                            {
                                aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                                aux1->info1.status = 'o';
                            }
                        }else{
                            aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                            aux1->info1.status = 'o';
                        }
                        //
                        if (aux1->Ninfos == 2)
                        {
                            if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_F)
                            {
                                aux2 = busca45(RAIZAvr, aux1->info2.Bloco_F +
1);
                                if (aux2->Ninfos == 2)
                                {
                                    if (aux2->info2.Bloco_I == aux1-
>info2.Bloco_F + 1)
                                    {
                                        aux1->info2.Bloco_F = aux2-
>info2.Bloco_F;
                                        aux1->info2.status = 'o';
                                        aux3 = aux2->info2;
                                    }
                                    else
                                    {
                                        aux1->info2.Bloco_F = aux2-
>info1.Bloco_F;
                                        aux1->info2.status = 'o';
                                        aux3 = aux2->info1;
                                    }
                                }
                                else
                                {
                                    aux1->info2.Bloco_F = aux2->info1.Bloco_F;
                                    aux1->info2.status = 'o';
                                    aux3 = aux2->info1;
                                }
                            }
                            else
                            {
                                aux2 = busca45(RAIZAvr, aux1->info1.Bloco_F +
1);
```

```
                                    if (aux2->Ninfos == 2)
                                    {
                                        if (aux2->info2.Bloco_I == aux1-
>info1.Bloco_F + 1)
                                        {
                                            aux1->info1.Bloco_F = aux2-
>info2.Bloco_F;
                                            aux1->info1.status = 'o';
                                            aux3 = aux2->info2;
                                        }
                                        else
                                        {
                                            aux1->info1.Bloco_F = aux2-
>info1.Bloco_F;
                                            aux1->info1.status = 'o';
                                            aux3 = aux2->info1;
                                        }
                                    }
                                    else
                                    {
                                        aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                                        aux1->info1.status = 'o';
                                        aux3 = aux2->info1;
                                    }
                                }
                            }
                            else
                            {
                                aux2 = busca45(RAIZAvr, aux1->info1.Bloco_F + 1);
                                if (aux2->Ninfos == 2)
                                {
                                    if (aux2->info2.Bloco_I == aux1->info1.Bloco_F
+ 1)
                                    {
                                        aux1->info1.Bloco_F = aux2->info2.Bloco_F;
                                        aux1->info1.status = 'o';
                                        aux3 = aux2->info2;
                                    }
                                    else
                                    {
                                        aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                                        aux1->info1.status = 'o';
                                        aux3 = aux2->info1;
                                    }
                                }
                                else
                                {
                                    aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                                    aux1->info1.status = 'o';
                                    aux3 = aux2->info1;
                                }
```

```
                    }
                    remover(RAIZAvr, NULL, (*raiz)->info1.Bloco_I);
                    //
                }
            }
        }else{
            aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_I - 1);
            if (aux1->Ninfos == 2)
            {
                if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_I - 1)
                {
                    aux1->info2.Bloco_F = (*raiz)->info1.Bloco_F;
                    aux1->info2.status = 'o';
                }
                else
                {
                    aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                    aux1->info1.status = 'o';
                }
            }
            else
            {
                aux1->info1.Bloco_F = (*raiz)->info1.Bloco_F;
                aux1->info1.status = 'o';
            }

            if (aux1->Ninfos == 2)
            {
                if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_F)
                {
                    aux2 = busca45(RAIZAvr, aux1->info2.Bloco_F + 1);
                    if (aux2->Ninfos == 2)
                    {
                        if (aux2->info2.Bloco_I == aux1->info2.Bloco_F + 1)
                        {
                            aux1->info2.Bloco_F = aux2->info2.Bloco_F;
                            aux1->info2.status = 'o';
                            aux3 = aux2->info2;
                        }
                        else
                        {
                            aux1->info2.Bloco_F = aux2->info1.Bloco_F;
                            aux1->info2.status = 'o';
                            aux3 = aux2->info1;
                        }
                    }
                    else
                    {
                        aux1->info2.Bloco_F = aux2->info1.Bloco_F;
                        aux1->info2.status = 'o';
                        aux3 = aux2->info1;
```

```
                    }
                }
                else
                {
                    aux2 = busca45(RAIZAvr, aux1->info1.Bloco_F + 1);
                    if (aux2->Ninfos == 2)
                    {
                        if (aux2->info2.Bloco_I == aux1->info1.Bloco_F + 1)
                        {
                            aux1->info1.Bloco_F = aux2->info2.Bloco_F;
                            aux1->info1.status = 'o';
                            aux3 = aux2->info2;
                        }
                        else
                        {
                            aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                            aux1->info1.status = 'o';
                            aux3 = aux2->info1;
                        }
                    }
                    else
                    {
                        aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                        aux1->info1.status = 'o';
                        aux3 = aux2->info1;
                    }
                }
            }
            else
            {
                aux2 = busca45(RAIZAvr, aux1->info1.Bloco_F + 1);
                if (aux2->Ninfos == 2)
                {
                    if (aux2->info2.Bloco_I == aux1->info1.Bloco_F + 1)
                    {
                        aux1->info1.Bloco_F = aux2->info2.Bloco_F;
                        aux1->info1.status = 'o';
                        aux3 = aux2->info2;
                    }
                    else
                    {
                        aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                        aux1->info1.status = 'o';
                        aux3 = aux2->info1;
                    }
                }
                else
                {
                    aux1->info1.Bloco_F = aux2->info1.Bloco_F;
                    aux1->info1.status = 'o';
                    aux3 = aux2->info1;
```

```
                    }
                }
                remover(RAIZAvr, NULL, (*raiz)->info1.Bloco_I);
            }
        }
    }else{
        if (*pai == NULL)
        {
            if ((*raiz)->Ninfos == 2)
            {
                if (ehFolha(*raiz))
                {
                    (*raiz)->info1.Bloco_F -= Qtd;
                    (*raiz)->info2.Bloco_I -= Qtd;
                    aux3.Bloco_I = -2;
                }
                else
                {
                    aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_F + 1);
                    (*raiz)->info1.Bloco_F -= Qtd;
                    aux1->info1.Bloco_I -= Qtd;
                    aux3.Bloco_I = -2;
                }
            }
            else
            {
                if (ehFolha(*raiz))
                {
                    (*raiz)->info1.Bloco_F -= Qtd;
                    (*raiz)->Ninfos = 2;
                    (*raiz)->info2.Bloco_I = (*raiz)->info1.Bloco_F + 1;
                    (*raiz)->info2.Bloco_F = Tam_bloco - 1;
                    (*raiz)->info2.status = 'o';
                    aux3.Bloco_I = -2;
                }
                else
                {
                    aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_F + 1);
                    (*raiz)->info1.Bloco_F -= Qtd;
                    aux1->info1.Bloco_I -= Qtd;
                    aux3.Bloco_I = -2;
                }
            }
        }
        else
        {
            if ((*raiz)->info1.Bloco_F == Tam_bloco - 1)
            {
                aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_I - 1);
                if (aux1->Ninfos == 2)
                {
```

```
                    if (aux1->info2.Bloco_F == (*raiz)->info1.Bloco_I - 1)
                    {
                        aux1->info2.Bloco_F += Qtd;
                        (*raiz)->info1.Bloco_I += Qtd;
                    }
                    else
                    {
                        aux1->info1.Bloco_F += Qtd;
                        (*raiz)->info1.Bloco_I += Qtd;
                    }
                }
                else
                {
                    aux1->info1.Bloco_F += Qtd;
                    (*raiz)->info1.Bloco_I += Qtd;
                }
                aux3.Bloco_I = -2;
            }
            else
            {
                aux1 = busca45(RAIZAvr, (*raiz)->info1.Bloco_F + 1);
                if (aux1->Ninfos == 2)
                {
                    if (aux1->info2.Bloco_I == (*raiz)->info1.Bloco_F + 1)
                    {
                        aux1->info2.Bloco_I -= Qtd;
                        (*raiz)->info1.Bloco_F -= Qtd;
                    }
                    else
                    {
                        aux1->info1.Bloco_I -= Qtd;
                        (*raiz)->info1.Bloco_F -= Qtd;
                    }
                }
                else
                {
                    aux1->info1.Bloco_I -= Qtd;
                    (*raiz)->info1.Bloco_F -= Qtd;
                }
                aux3.Bloco_I = -2;
            }
        }
    }
}else if ((*raiz)->Ninfos == 2){
    espaco = ((*raiz)->info2.Bloco_F - (*raiz)->info2.Bloco_I) + 1;
    if (espaco >= Qtd && (*raiz)->info2.status == 'l')
    {
        printf("\n---Bloco que foi Ocupado---\n");
        printf("  -Inicio: %d\n  -Final: %d\n  -Status: %c\n", (*raiz)-
>info2.Bloco_I, (*raiz)->info2.Bloco_F, (*raiz)->info2.status);
        aux = 1;
```

```
            if (espaco == Qtd)
            {
                if (*pai == NULL)
                {
                    if (ehFolha(*raiz))
                    {
                        (*raiz)->info1.Bloco_F += Qtd;
                        (*raiz)->Ninfos = 1;
                        aux3.Bloco_I = -2;
                    }
                    else
                    {
                        aux1 = busca45(RAIZAvr, (*raiz)->info2.Bloco_F + 1);
                        if (aux1->Ninfos == 1)
                        {
                            aux2 = busca45(RAIZAvr, (*raiz)->info2.Bloco_I - 1);
                            if (aux2->Ninfos == 1)
                            {
                                aux1->info1.Bloco_I = aux2->info1.Bloco_I;
                                aux1->info1.status = 'o';
                                aux3 = aux2->info1;
                            }
                            else
                            {
                                aux1->info1.Bloco_I = aux2->info2.Bloco_I;
                                aux1->info1.status = 'o';
                                aux3 = aux2->info2;
                            }
                        }
                        else
                        {
                            aux2 = busca45(RAIZAvr, (*raiz)->info2.Bloco_I - 1);
                            if (aux2->Ninfos == 1)
                            {
                                aux1->info2.Bloco_I = aux2->info1.Bloco_I;
                                aux1->info2.status = 'o';
                                aux3 = aux2->info1;
                            }
                            else
                            {
                                aux1->info2.Bloco_I = aux2->info2.Bloco_I;
                                aux1->info2.status = 'o';
                                aux3 = aux2->info2;
                            }
                        }
                        remover(RAIZAvr, NULL, (*raiz)->info2.Bloco_I);
                    }
                }
                else
                {
                    if (ehFolha(*raiz))
```

```
                {
                    (*raiz)->info1.Bloco_F += Qtd;
                    (*raiz)->Ninfos = 1;
                    aux3.Bloco_I = -2;
                }
                else
                {
                    aux1 = busca45(RAIZAvr, (*raiz)->info2.Bloco_F + 1);
                    if (aux1->Ninfos == 1)
                    {
                        aux2 = busca45(RAIZAvr, (*raiz)->info2.Bloco_I - 1);
                        if (aux2->Ninfos == 1)
                        {
                            aux1->info1.Bloco_I = aux2->info1.Bloco_I;
                            aux1->info1.status = 'o';
                            aux3 = aux2->info1;
                        }
                        else
                        {
                            aux1->info1.Bloco_I = aux2->info2.Bloco_I;
                            aux1->info1.status = 'o';
                            aux3 = aux2->info2;
                        }
                    }
                    else
                    {
                        aux2 = busca45(RAIZAvr, (*raiz)->info2.Bloco_I - 1);
                        if (aux2->Ninfos == 1)
                        {
                            aux1->info2.Bloco_I = aux2->info1.Bloco_I;
                            aux1->info2.status = 'o';
                            aux3 = aux2->info1;
                        }
                        else
                        {
                            aux1->info2.Bloco_I = aux2->info2.Bloco_I;
                            aux1->info2.status = 'o';
                            aux3 = aux2->info2;
                        }
                    }
                    remover(RAIZAvr, NULL, (*raiz)->info2.Bloco_I);
                }
            }
        }
        else
        {
            if (*pai == NULL)
            {
                if (ehFolha(*raiz))
                {
                    (*raiz)->info1.Bloco_F += Qtd;
```

```
                    (*raiz)->info2.Bloco_I += Qtd;
                    aux3.Bloco_I = -2;
                }
                else
                {
                    aux1 = busca45(RAIZAvr, (*raiz)->info2.Bloco_F + 1);
                    if (aux1->Ninfos == 1)
                    {
                        (*raiz)->info2.Bloco_F -= Qtd;
                        aux1->info1.Bloco_I -= Qtd;
                        aux3.Bloco_I = -2;
                    }
                    else
                    {
                        (*raiz)->info2.Bloco_F -= Qtd;
                        aux1->info2.Bloco_I -= Qtd;
                        aux3.Bloco_I = -2;
                    }
                }
            }
            else
            {
                if (ehFolha(*raiz))
                {
                    (*raiz)->info1.Bloco_F += Qtd;
                    (*raiz)->info2.Bloco_I += Qtd;
                    aux3.Bloco_I = -2;
                }
                else
                {
                    aux1 = busca45(RAIZAvr, (*raiz)->info2.Bloco_F + 1);
                    if (aux1->Ninfos == 1)
                    {
                        (*raiz)->info2.Bloco_F -= Qtd;
                        aux1->info1.Bloco_I -= Qtd;
                        aux3.Bloco_I = -2;
                    }
                    else
                    {
                        (*raiz)->info2.Bloco_F -= Qtd;
                        aux1->info2.Bloco_I -= Qtd;
                        aux3.Bloco_I = -2;
                    }
                }
            }
        }
    }
    if (aux == 0)
    { // percorre a árvore
        aux3 = ProcuraEspaco(&(*raiz)->Esq, Qtd, raiz, RAIZAvr, ptr);
        if (aux3.Bloco_I == -1)
```

```
            {
                aux3 = ProcuraEspaco(&(*raiz)->Cen_Esq, Qtd, raiz, RAIZAvr, ptr);
            }
            if (aux3.Bloco_I == -1)
            {
                aux3 = ProcuraEspaco(&(*raiz)->Cen, Qtd, raiz, RAIZAvr, ptr);
            }
            if (aux3.Bloco_I == -1)
            {
                aux3 = ProcuraEspaco(&(*raiz)->Cen_Dir, Qtd, raiz, RAIZAvr, ptr);
            }
            if (aux3.Bloco_I == -1)
            {
                aux3 = ProcuraEspaco(&(*raiz)->Dir, Qtd, raiz, RAIZAvr, ptr);
            }
            // Adapte para a quantidade de filhos da árvore 2-3-4-5
        }
    }
    return aux3;
}

Arv45 *busca45(Arv45 **raiz, int info)
{
    Arv45 *aux = NULL;
    if (*raiz != NULL)
    {
        if ((info >= (*raiz)->info1.Bloco_I && info <= (*raiz)->info1.Bloco_F) ||
            (info >= (*raiz)->info2.Bloco_I && info <= (*raiz)->info2.Bloco_F) ||
            (info >= (*raiz)->info3.Bloco_I && info <= (*raiz)->info3.Bloco_F) ||
            (info >= (*raiz)->info4.Bloco_I && info <= (*raiz)->info4.Bloco_F))
        {
            aux = *raiz;
        }
        else if (info < (*raiz)->info1.Bloco_I)
        {
            aux = busca45(&(*raiz)->Esq, info);
        }
        else if ((info > (*raiz)->info1.Bloco_F && info < (*raiz)->info2.Bloco_I)
|| (*raiz)->Ninfos == 1)
        {
            aux = busca45(&(*raiz)->Cen_Esq, info);
        }
        else if ((info > (*raiz)->info2.Bloco_F && info < (*raiz)->info3.Bloco_I)
|| (*raiz)->Ninfos == 2)
        {
            aux = busca45(&(*raiz)->Cen, info);
        }
        else if ((info > (*raiz)->info3.Bloco_F && info < (*raiz)->info4.Bloco_I)
|| (*raiz)->Ninfos == 3)
        {
            aux = busca45(&(*raiz)->Cen_Dir, info);
```

```
        }
        else if (info > (*raiz)->info4.Bloco_F && (*raiz)->Ninfos == 4)
        {
            aux = busca45(&(*raiz)->Dir, info);
        }
    }
    return aux;
}

int remover(Arv45 **raiz, Arv45 **pai, int valor){
    int removeu = 0;
    Arv45 *noMenorInfoDir = NULL;
    if (*raiz != NULL)
    {
        if (valor == (*raiz)->info1.Bloco_I || valor == (*raiz)->info2.Bloco_I)
        {
            if (valor == (*raiz)->info1.Bloco_I)
            {
                if (ehFolha(*raiz))
                {
                    removeu = removeInfo1Folha(*raiz, *pai);
                    if (removeu == 3)
                    {
                        free(*raiz);
                        *raiz = NULL;
                    }
                }
                else
                {
                    removeu = removeInfo1NaoFolha((*raiz));
                }
            }
            else
            {
                if (ehFolha(*raiz))
                { // se for folha, so remove o no
                    removeu = removeInfo2Folha(*raiz);
                }
                else if (ehFolha((*raiz)->Dir))
                {
                    removeu = removeInfo2ComDirFolha((*raiz)->Dir, *raiz);
                }
                else
                {
                    if (numeroInfosArv((*raiz)->Dir) > 3)
                    {
                        noMenorInfoDir = obterMenorNo((*raiz)->Dir);
                        (*raiz)->info2 = noMenorInfoDir->info1;
                        removeu = remover(&((*raiz)->Dir), raiz, noMenorInfoDir-
>info1.Bloco_I);
                    }
```

```
                    else
                    {
                        removeu = removeInfo2NaoFolha(*raiz);
                    }
                }
            }
        }
        else if (valor < (*raiz)->info1.Bloco_I)
        {
            removeu = remover(&((*raiz)->Esq), raiz, valor);
        }
        else if (valor < (*raiz)->info2.Bloco_I || (*raiz)->Ninfos == 1)
        {
            removeu = remover(&((*raiz)->Cen), raiz, valor);
        }
        else if (valor > (*raiz)->info2.Bloco_I && (*raiz)->Ninfos == 2)
        {
            removeu = remover(&((*raiz)->Dir), raiz, valor);
        }
        // balancear;
        if (removeu == 2)
        {
            removeu = balancear(&(*raiz), &(*pai), removeu);
        }
    }
    return removeu;
}

int balancear(Arv45 **raiz, Arv45 **pai, int removeu)
{
    if ((*pai) == NULL)
    {
        (*raiz) = (*raiz)->Esq;
        removeu = 1;
    }
    else
    {
        if ((*pai)->Esq == (*raiz))
        {
            if ((*pai)->Cen->Ninfos == 2)
            {
                (*raiz)->info1 = (*pai)->info1;
                (*raiz)->Cen = (*pai)->Cen->Esq;
                (*pai)->info1 = (*pai)->Cen->info1;
                (*pai)->Cen->info1 = (*pai)->Cen->info2;
                (*pai)->Cen->Ninfos = 1;
                (*pai)->Cen->Esq = (*pai)->Cen->Cen;
                (*pai)->Cen->Cen = (*pai)->Cen->Dir;
                (*pai)->Cen->Dir = NULL;
                removeu = 1;
            }
```

```
            else
            {
                if ((*pai)->Ninfos == 2)
                {
                    (*raiz)->info1 = (*pai)->info1;
                    (*raiz)->info2 = (*pai)->Cen->info1;
                    (*raiz)->Ninfos = 2;
                    (*raiz)->Cen = (*pai)->Cen->Esq;
                    (*raiz)->Dir = (*pai)->Cen->Cen;
                    free((*pai)->Cen);
                    (*pai)->info1 = (*pai)->info2;
                    (*pai)->Ninfos = 1;
                    (*pai)->Cen = (*pai)->Dir;
                    (*pai)->Dir = NULL;
                    removeu = 1;
                }
                else
                {
                    (*raiz)->info1 = (*pai)->info1;
                    (*raiz)->info2 = (*pai)->Cen->info1;
                    (*raiz)->Ninfos = 2;
                    (*raiz)->Cen = (*pai)->Cen->Esq;
                    (*raiz)->Dir = (*pai)->Cen->Cen;
                    free((*pai)->Cen);
                    (*pai)->Cen = NULL;
                }
            }
        }
        else if ((*pai)->Cen == (*raiz))
        {
            if ((*pai)->Esq->Ninfos == 2)
            {
                (*raiz)->info1 = (*pai)->info1;
                (*pai)->info1 = (*pai)->Esq->info2;
                (*raiz)->Cen = (*raiz)->Esq;
                (*raiz)->Esq = (*pai)->Esq->Dir;
                (*pai)->Esq->Dir = NULL;
                (*pai)->Esq->Ninfos = 1;
                removeu = 1;
            }
            else
            {
                if ((*pai)->Ninfos == 2)
                {
                    (*pai)->Esq->info2 = (*pai)->info1;
                    (*pai)->Esq->Ninfos = 2;
                    (*pai)->Esq->Dir = (*raiz)->Esq;
                    free(*raiz);
                    *raiz = NULL;
                    free((*pai)->Cen);
                    (*pai)->Cen = NULL;
```

```
                    (*pai)->info1 = (*pai)->info2;
                    (*pai)->Ninfos = 1;
                    (*pai)->Cen = (*pai)->Dir;
                    (*pai)->Dir = NULL;
                    removeu = 1;
                }
                else
                {
                    (*pai)->Esq->info2 = (*pai)->info1;
                    (*pai)->Esq->Ninfos = 2;
                    (*pai)->Esq->Dir = (*pai)->Cen->Esq;
                    free((*pai)->Cen);
                    (*pai)->Cen = NULL;
                }
            }
        }
        else
        {
            if ((*pai)->Cen->Ninfos == 2)
            {
                (*raiz)->info1 = (*pai)->info2;
                (*raiz)->Cen = (*raiz)->Esq;
                (*raiz)->Esq = (*pai)->Cen->Dir;
                (*pai)->info2 = (*pai)->Cen->info2;
                (*pai)->Cen->Ninfos = 1;
                (*pai)->Cen->Dir = NULL;
            }
            else
            {
                (*pai)->Cen->info2 = (*pai)->info2;
                (*pai)->Cen->Ninfos = 2;
                (*pai)->Cen->Dir = (*raiz)->Esq;
                free(*raiz);
                *raiz = NULL;
                (*pai)->Ninfos = 1;
                free((*pai)->Dir);
                (*pai)->Dir = NULL;
            }
            removeu = 1;
        }
    }
    return removeu;
}

int removeInfo2Folha(Arv45 *raiz)
{
    raiz->Ninfos = 1;
    return 1;
}

int removeInfo2ComDirFolha(Arv45 *raiz, Arv45 *pai)
```

```
{
    int removeu = 0;
    if (raiz->Ninfos == 2)
    {
        pai->info2 = raiz->info1;
        raiz->info1 = raiz->info2;
        raiz->Ninfos = 1;
        removeu = 1;
    }
    else
    {
        if (pai->Cen->Ninfos == 2)
        {
            pai->info2 = pai->Cen->info2;
            pai->Cen->Ninfos = 1;
            removeu = 1;
        }
        else
        {
            pai->Cen->info2 = raiz->info1;
            pai->Cen->Ninfos = 2;
            pai->Ninfos = 1;
            free(raiz);
            pai->Dir = NULL;
            removeu = 1;
        }
    }
    return removeu;
}

Arv45 *obterMenorNo(Arv45 *raiz)
{
    Arv45 *aux = NULL;
    aux = raiz;
    if (raiz->Esq != NULL)
    {
        aux = obterMenorNo(raiz->Esq);
    }
    return aux;
}

int numeroInfosArv(Arv45 *raiz)
{
    int cont = 0;
    if (raiz != NULL)
    {
        cont += 1;
        if (raiz->Ninfos == 2)
        {
            cont += 1;
        }
```

```
        cont += numeroInfosArv(raiz->Esq);
        cont += numeroInfosArv(raiz->Cen);
        cont += numeroInfosArv(raiz->Dir);
    }
    return cont;
}

Arv45 *obterMaiorNo(Arv45 *raiz)
{
    Arv45 *aux = NULL;
    aux = raiz;
    if (raiz->Dir != NULL)
    {
        aux = obterMaiorNo(raiz->Dir);
    }
    else
    {
        if (raiz->Cen != NULL)
        {
            aux = obterMaiorNo(raiz->Cen);
        }
    }
    return aux;
}

int removeInfo2NaoFolha(Arv45 *raiz)
{
    int removeu = 0;
    Arv45 *noMaiorInfoCen = NULL;
    noMaiorInfoCen = obterMaiorNo(raiz->Cen);

    if (noMaiorInfoCen->Ninfos == 2)
    {
        raiz->info2 = noMaiorInfoCen->info2;
        removeu = remover(&(raiz->Cen), &raiz, noMaiorInfoCen->info2.Bloco_I);
    }
    else
    {
        raiz->info2 = noMaiorInfoCen->info1;
        removeu = remover(&(raiz->Cen), &raiz, noMaiorInfoCen->info1.Bloco_I);
    }
    return removeu;
}

int removeInfo1Folha(Arv45 *raiz, Arv45 *pai)
{
    int removeu = 0;
    if (raiz->Ninfos == 2)
    {
        raiz->info1 = raiz->info2;
        raiz->Ninfos = 1;
```

```
        removeu = 1;
    }
    else if (pai == NULL)
    {
        removeu = 3;
    }
    else
    {
        if (pai->Esq == raiz)
        {
            if (pai->Cen->Ninfos == 2)
            {
                raiz->info1 = pai->info1;
                pai->info1 = pai->Cen->info1;
                pai->Cen->info1 = pai->Cen->info2;
                pai->Cen->Ninfos = 1;
                removeu = 1;
            }
            else if (pai->Ninfos == 2)
            {
                raiz->info1 = pai->info1;
                raiz->info2 = pai->Cen->info1;
                raiz->Ninfos = 2;
                free(pai->Cen);
                pai->info1 = pai->info2;
                pai->Ninfos = 1;
                pai->Cen = pai->Dir;
                pai->Dir = NULL;
                removeu = 1;
            }
            else
            {
                removeu = 2;
            }
        }
        else if (pai->Cen == raiz)
        {
            if (pai->Esq->Ninfos == 2)
            {
                raiz->info1 = pai->info1;
                pai->info1 = pai->Esq->info2;
                pai->Esq->Ninfos = 1;
                removeu = 1;
            }
            else if (pai->Ninfos == 2)
            {
                if (pai->Dir->Ninfos == 2)
                {
                    raiz->info1 = pai->info2;
                    pai->info2 = pai->Dir->info1;
                    pai->Dir->info1 = pai->Dir->info2;
```

```
                    pai->Dir->Ninfos = 1;
                    removeu = 1;
                }
                else
                {
                    raiz->info1 = pai->info2;
                    raiz->info2 = pai->Dir->info1;
                    raiz->Ninfos = 2;
                    pai->Ninfos = 1;
                    free(pai->Dir);
                    pai->Dir = NULL;
                    removeu = 1;
                }
            }
            else
            {
                removeu = 2;
            }
        }
        else
        {
            if (pai->Cen->Ninfos == 2)
            {
                raiz->info1 = pai->info2;
                pai->info2 = pai->Cen->info2;
                pai->Cen->Ninfos = 1;
                removeu = 1;
            }
            else
            {
                pai->Cen->info2 = pai->info2;
                pai->Cen->Ninfos = 2;
                pai->Ninfos = 1;
                free(pai->Dir);
                pai->Dir = NULL;
                removeu = 1;
            }
        }
    }
    return removeu;
}

int removeInfo1NaoFolha(Arv45 *raiz)
{
    int removeu = 0;
    Arv45 *noMaiorInfoEsq = NULL;
    noMaiorInfoEsq = obterMaiorNo(raiz->Esq);

    if (noMaiorInfoEsq->Ninfos == 2)
    {
        raiz->info1 = noMaiorInfoEsq->info2;
```

```
        removeu = remover(&(raiz->Esq), &raiz, noMaiorInfoEsq->info2.Bloco_I);
    }
    else
    {
        raiz->info1 = noMaiorInfoEsq->info1;
        removeu = remover(&(raiz->Esq), &raiz, noMaiorInfoEsq->info1.Bloco_I);
    }
    return removeu;
}

void liberarMemoriaArvore(Arv45 *raiz) {
    if (raiz != NULL) {
        liberarMemoriaArvore(raiz->Esq);
        liberarMemoriaArvore(raiz->Cen_Esq);
        liberarMemoriaArvore(raiz->Cen);
        liberarMemoriaArvore(raiz->Cen_Dir);
        liberarMemoriaArvore(raiz->Dir);
        free(raiz);
    }
}
```