Ministério da Educação
Universidade Federal do Piauí – UFPI
Campus Senador Helvídio Nunes de Barros – Picos
Curso Bacharelado em Sistemas de Informação
Professora: Juliana Oliveira de Carvalho
Discente: Crisly Maria Silva dos Santos
Disciplina: Estrutura de Dados II– 2023.2

Relatório Trabalho II

Resumo do projeto

Este relatório tem como objetivo ilustrar a implementação prática das estruturas de dados Árvore Binária Rubro-Negra, Árvore 2-3 e Árvore 4-5, utilizando a linguagem de programação C. Foi elaborado para resolver uma série de problemas práticos propostos. O relatório detalha cada etapa do processo de resolução dos problemas, apresentando de forma clara e coerente as implementações realizadas. Além disso, oferece uma visão abrangente e significativa das aplicações dessas estruturas de dados.

Introdução

Nesta seção será apresentado a Introdução do projeto. Estruturas de dados são fundamentais para a programação e muito se fala sobre a importância do estudo desses temas. Elas representam 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. Basicamente, estruturas de dados são objetos que contêm dados, na forma de campos, muitas vezes conhecidos como atributos; e código, na forma de procedimentos, muitas vezes conhecidos como métodos.

Uma Árvore binária rubro-negra é um tipo de árvore binária de busca balanceada, usada em ciência da computação, tipicamente para implementar vetores associativos. A estrutura original foi inventada em 1972 e é complexa, mas tem um bom pior-caso de tempo de execução para suas operações e é eficiente na prática: buscar, inserir e remover.

Uma Árvore 2-3 é uma estrutura de dados na qual cada nó pode conter até duas informações e ter, no máximo, três filhos. Uma característica distintiva dessa estrutura é que todas as inserções são realizadas exclusivamente nos nós folhas. Por outro lado, a Árvore 4-5 é uma estrutura mais complexa, permitindo até quatro informações em um único nó e tendo a capacidade de acomodar até cinco

filhos. Semelhante à Árvore 2-3, a Árvore 4-5 também realiza inserções apenas nos nós folhas. Essas estruturas de árvore oferecem eficiência na busca de informações, mantendo os dados ordenados e balanceados, facilitando operações como inserção, busca e remoção.

Na continuação do projeto tem como objetivo mostrar ao leitor experimentos e justificativas acerca dos problemas resolvidos nessa estrutura de dados. A seção Seções Específicas mostrará os aspectos funcionais presentes no programa, a seção Resultados da Execução do Programa mostrará resultados através da execução do programa, bem como se familiarizar-se com o propósito do mesmo. A seção Conclusão reforça a informação já apresentada no projeto, bem como apresenta resumidamente o experimento. E por fim, a seção Apêndice contém os códigos fontes que foram produzidos como parte do experimento.

Seções Específicas

Nesta seção será apresentada as Seções Específicas. Neste trabalho foram fornecidos três problemas. O primeiro problema tem como propósito criar um programa em C de uma biblioteca de músicas. O programa deve ser projetado para permitir a inserção de artistas, álbuns e músicas. Isso significa que uma música deve estar vinculada a um álbum, e um álbum deve estar associado a um artista. Portanto, uma música só pode ser adicionada a um álbum previamente cadastrado, e um álbum só pode ser vinculado a um artista já existente no sistema.

O programa deve oferecer funcionalidades de busca abrangentes, permitindo pesquisas por artista, álbum e música. Além disso, deve ser possível remover uma música específica ou um álbum inteiro. Finalmente, a questão sugere a realização de um experimento que envolve a busca por 30 itens (neste caso, artistas). O programa deve ser capaz de exibir o caminho percorrido na árvore de dados para localizar o item e o tempo gasto nessa operação.

O segundo problema compartilha o mesmo objetivo que o primeiro. No entanto, em vez de empregar a estrutura de dados da Árvore Rubro-Negra, este problema requer o uso da estrutura de dados da Árvore 2-3. A natureza do problema permanece a mesma.

O terceiro problema propõe o seguinte, suponha que uma memória seja dividida em blocos lógicos de 1Mbyte, e que o primeiro bloco é o bloco 0, suponha também que o gerenciador de memória de um Sistema Operacional mantenha uma árvore 4-5 com nós para blocos livres e ocupados da memória. Cada nó da árvore 4-5 contém os seguintes campos: O- para ocupado ou L- para livre, o número do bloco inicial, o número do bloco final, endereço inicial (correspondente ao endereço inicial do primeiro bloco do nó) e endereço final

(correspondente ao endereço final do último bloco do nó). A questão é dividida em três partes a serem realizadas.

Letra A: Fazer um programa em C que cadastra os nós da árvore, onde o usuário deve informar se o primeiro nó é livre ou ocupado, o endereço inicial e final do nó. Os demais nós serão contabilizados pelo sistema se são livres ou ocupados e o usuário deve apenas informar o endereço final de cada um. O cadastro termina quando o usuário informar como endereço final de um nó o último endereço da memória.

Letra B: Fazer uma função que o usuário informe a quantidade de nós que ele precisa alocar e retorne as informações do nó que atenda às necessidades do usuário e então modifique a situação do referido nó de Livre para Ocupado.

(i) A árvore deve manter nós intercalados de acordo com a situação do nó, ou seja, se a situação de um nó muda então os nós adjacentes a ele deve ser concatenado. Consequentemente os nós da árvore serão modificados.
(ii) O nó escolhido possui uma quantidade maior de blocos do que o solicitado pelo usuário os nós árvore deve ser atualizados de forma que mantenha blocos adjacentes livres ou ocupados em um mesmo nó.

Letra C: Fazer uma função em que o usuário informe blocos que devem ser liberados.

(i) A árvore deve manter nós intercalados de acordo com a situação do nó, ou seja, se a situação de um nó muda então os nós adjacentes a ele deve ser concatenado. Consequentemente os nós da árvore serão modificados.
(ii) O nó escolhido possui uma quantidade maior de blocos do que o solicitado pelo usuário os nós árvore deve ser atualizados de forma que mantenha blocos adjacentes livres ou ocupados em um mesmo nó.

Resultados da Execução do Programa

Nesta seção será apresentado os resultados da execução dos programas realizados neste trabalho. Em relação à primeira questão, os elementos estão sendo inseridos corretamente, seguindo as regras de inserção da árvore rubro-negra. Um nó é inserido como vermelho, e as rotações necessárias são realizadas de acordo com a estrutura da árvore no momento da inserção e do balanceamento. A figura 01 mostra o menu desta questão.

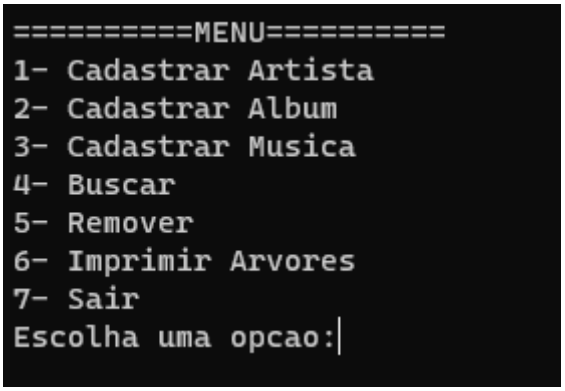

Figura 01 – Menu da Questão

Na opção 1 cadastrar artista, será solicitado ao usuário que ele informe o nome do artista e o estilo musical, para que assim prossiga com o cadastramento do artista. Na figura 02 mostra o teste realizado para a demonstração do primeiro caso.

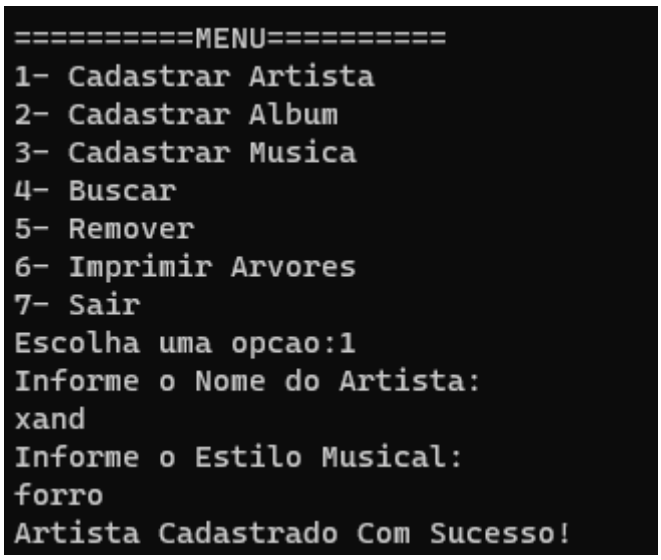

Figura 02 – Cadastro Artista

O próximo teste a ser realizado ainda no cadastro de artista, é quando o usuário tentar cadastrar um artista já existente. A figura 2.1 mostra a demonstração do teste feito.

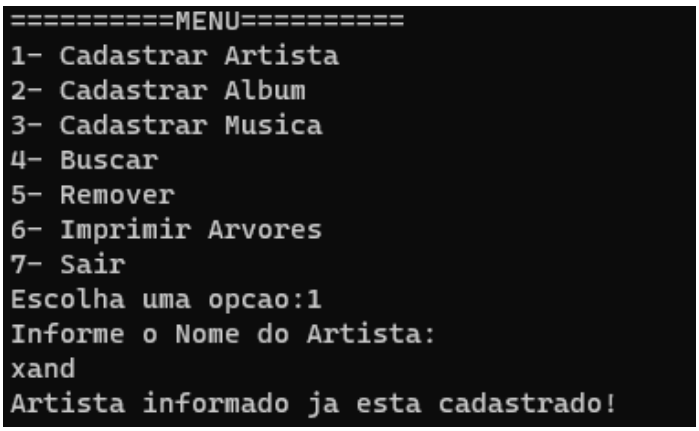

Figura 2.1

A partir do teste realizado, o programa garante que o usuário não consiga cadastrar um artista que já está registrado na sua árvore. O próximo teste a ser feito é na opção 2, cadastro de álbuns.

Na seção de registro de álbuns, o primeiro teste é realizado quando o usuário tenta registrar um álbum, mas o artista correspondente não está registrado. Portanto, não é possível registrar um álbum de um artista inexistente. Isso é ilustrado na Figura 03.

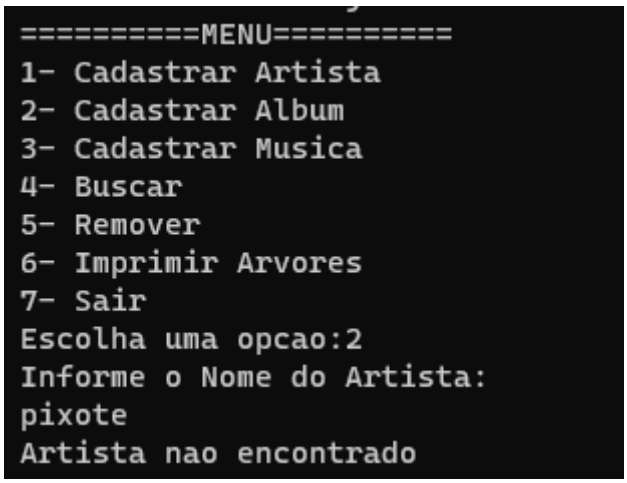

Figura 03

O teste subsequente ocorre quando o artista do álbum a ser registrado já existe na árvore. Assim, torna-se possível registrar o álbum na árvore. Isso é demonstrado na Figura 3.1.

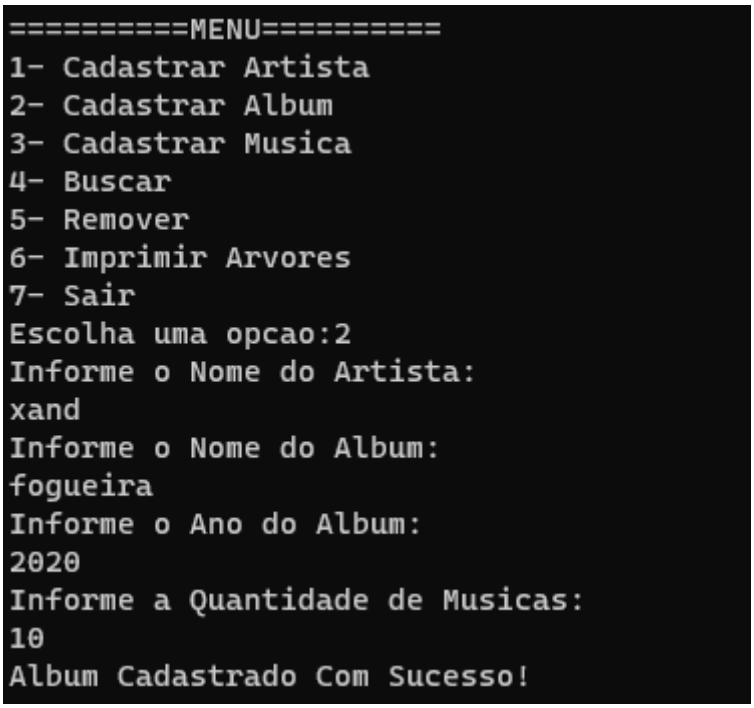

Figura 3.1

O teste seguinte, ainda na opção de registro de álbuns, ocorre quando o usuário tenta registrar um álbum que já está registrado na árvore. Nesse caso, o usuário não conseguirá realizar o registro. A figura 3.2 mostra isso.

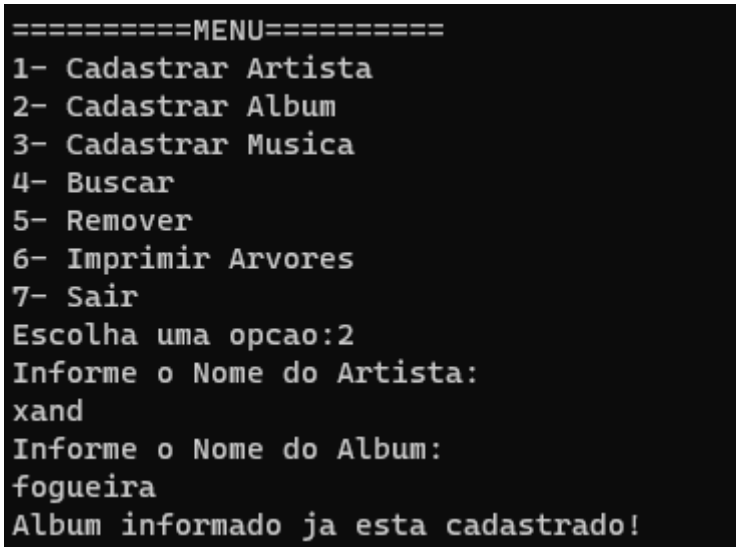

Figura 03.2

Com base nisso, pode-se observar que o programa está funcionando conforme o esperado e de acordo com o que foi solicitado na questão proposta. Está cumprindo todas as restrições e atendendo a todas as solicitações feitas pelo usuário.

A próxima opção a ser avaliada é a opção 3, que tem como objetivo o cadastro de músicas. O primeiro teste realizado com essa opção envolve a tentativa de cadastrar uma música para um artista que não está registrado. A figura 4 apresenta o teste feito.

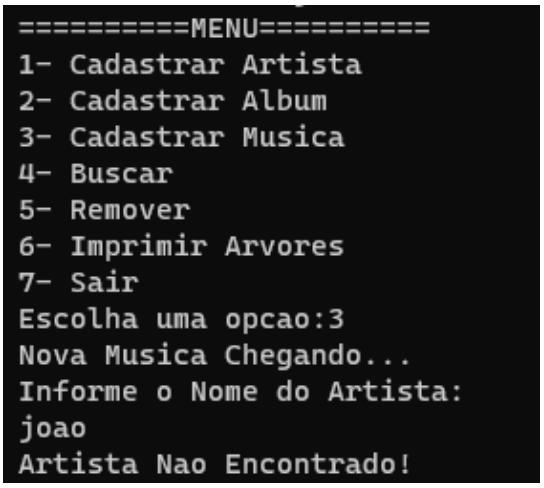

Figura 04

Observa-se que o programa não permite que o usuário cadastre uma música de um artista que não existe.

Continuando com a opção 3, o próximo teste a ser realizado é quando o usuário tenta cadastrar uma música em um álbum que não existe. Em outras palavras, embora o artista esteja registrado, o álbum não está tornando impossível cadastrar uma música em um álbum inexistente. Observe na figura 4.1.

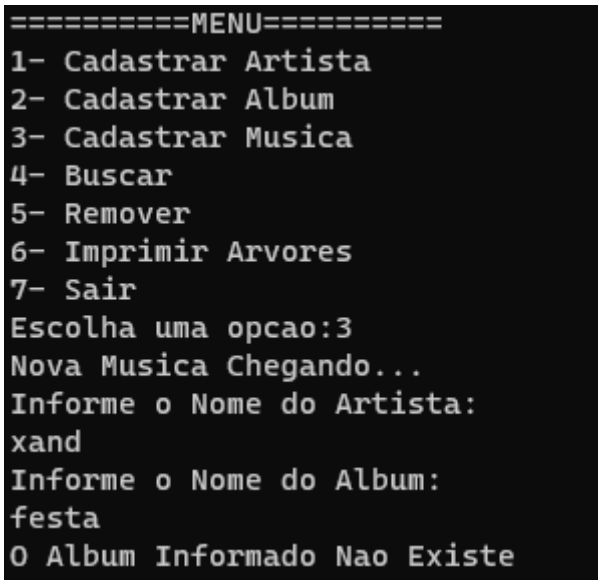

Figura 04.1

Agora o teste é realizado quando tanto o artista quanto o álbum estão registrados. Nesse caso, é possível cadastrar a música com sucesso. Isso é demonstrado na Figura 04.2.

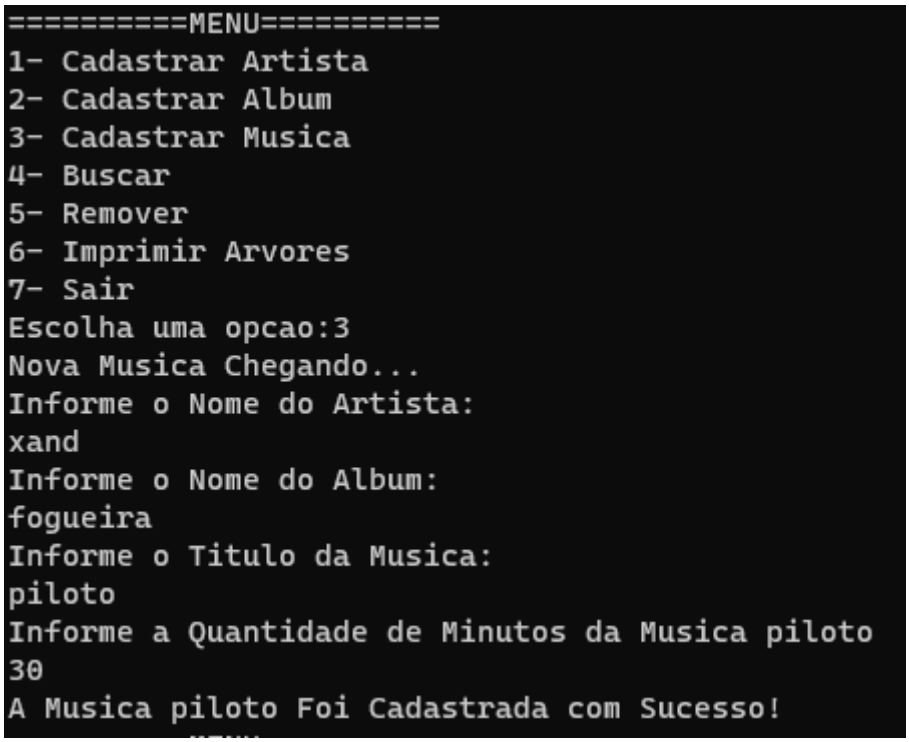

Figura 04.2

Finalmente, outro teste é realizado quando se tenta cadastrar uma música que já existe nesse álbum. Isso pode ser observado na Figura 4.3.

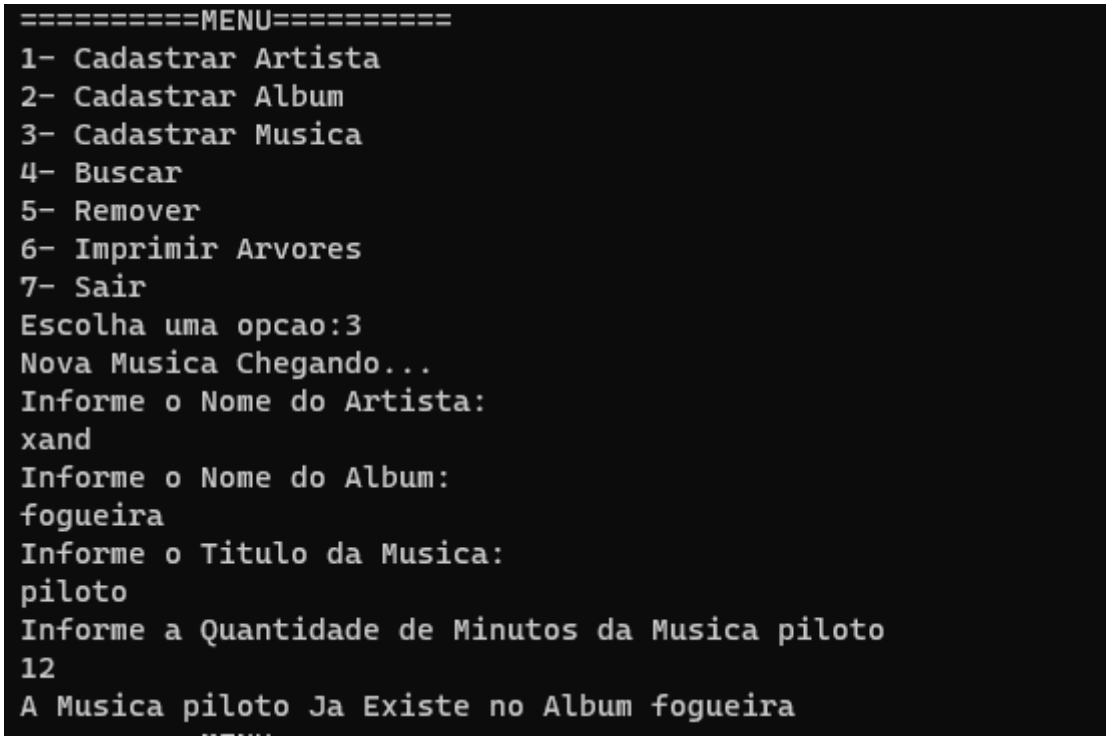

Figura 04.3

Após testar e demonstrar todas as possíveis entradas do usuário nas opções 1, 2 e 3, prosseguiremos agora para testar a opção 4, que se refere à função de busca. Isso inclui buscar por artista, álbum e música. O primeiro teste a ser realizado é a busca por um artista que já está cadastrado. Observe a figura 05.

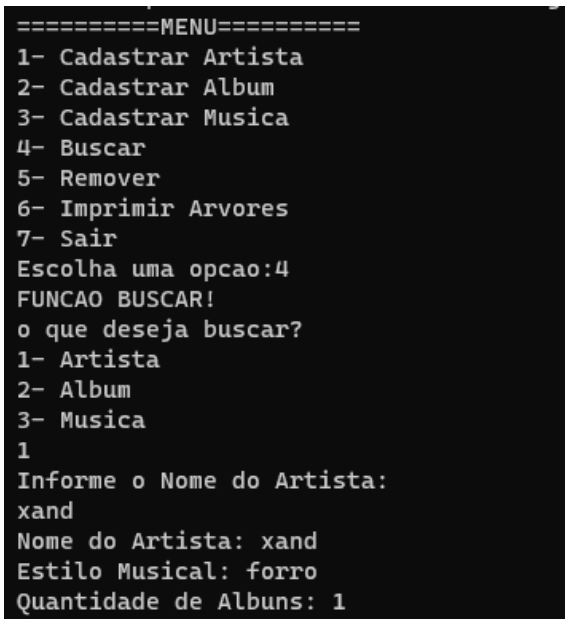

Figura 05

O próximo teste a ser feito ainda nessa opção é quando o usuário tenta buscar uma artista, mas esse artista não existe. A figura 5.1 mostra isso.

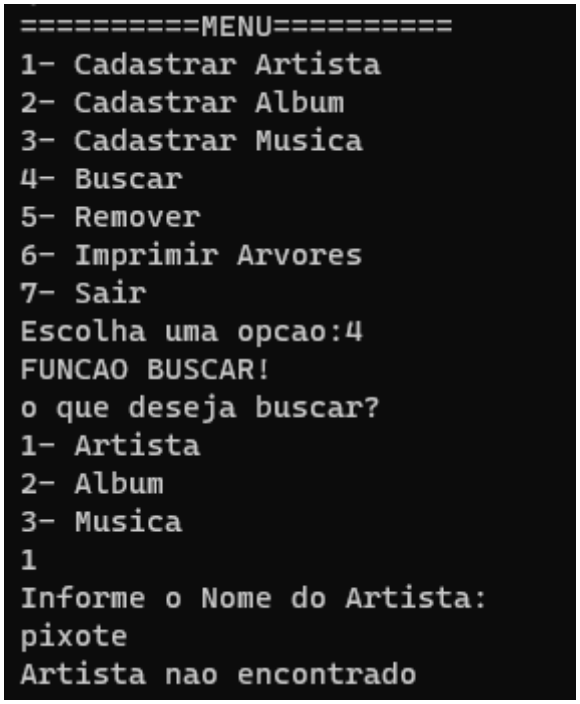

Figura 05.1

Depois dos testes feitos na opção 1 da função buscar, seguiremos a fazer o teste da próxima opção que é a de álbum. O primeiro teste a ser feito é o usuário buscar um álbum que já existe. Observe a figura 06.

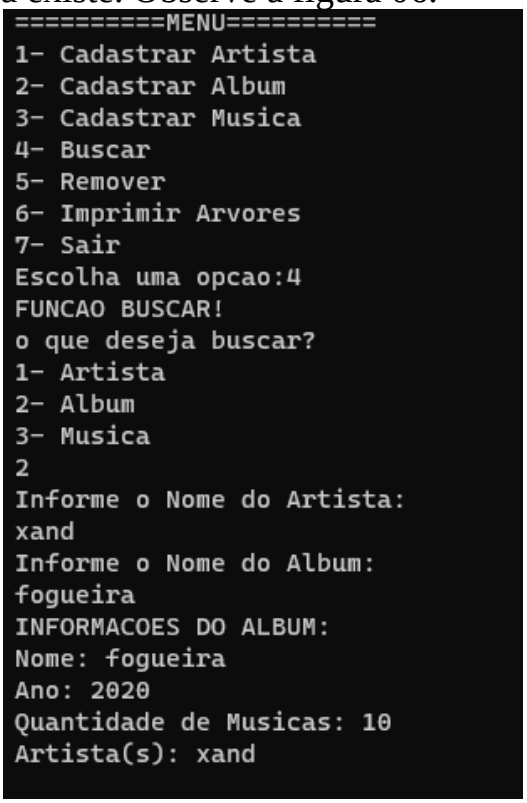

Figura 06

O próximo teste é quando o usuário tenta buscar um álbum, porém esse álbum não existe. Observe a demonstração na figura 6.1.

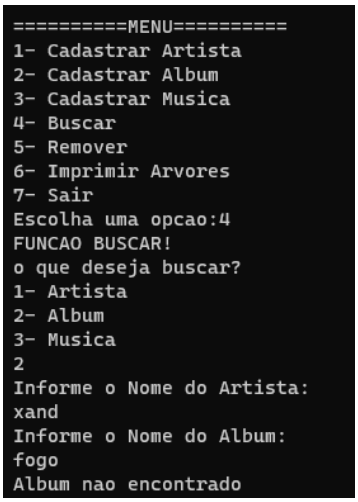

Figura 06.1

O próximo passo a ser feito é testar a opção 3 da função de buscar. Ou seja, buscar uma música. Primeiro experimento feito é buscar uma música que já existe. Observe a figura 7.

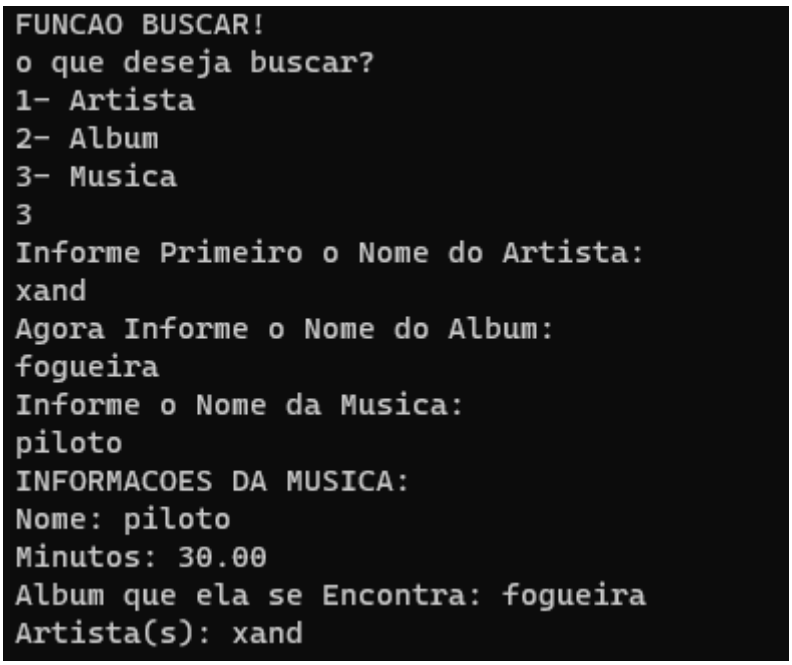

Figura 07.

Agora o próximo teste a ser feito é quando o usuário tentar buscar uma música de um artista que não existe. Observe na figura 7.1.

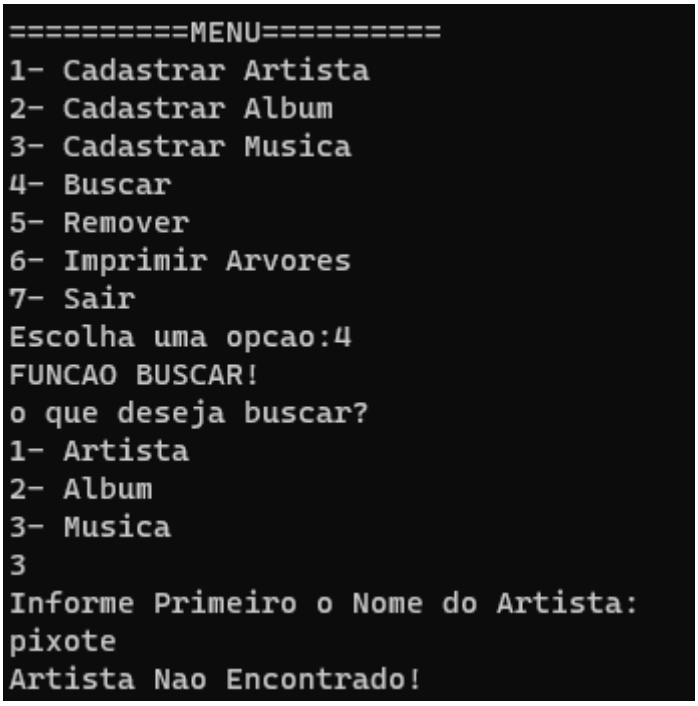

Figura 07.1

Por fim, para finalizar essa parte de busca de música o último teste feito é quando o usuário tentar buscar uma música, porém o álbum informado não existe. Observe na figura 07.2.

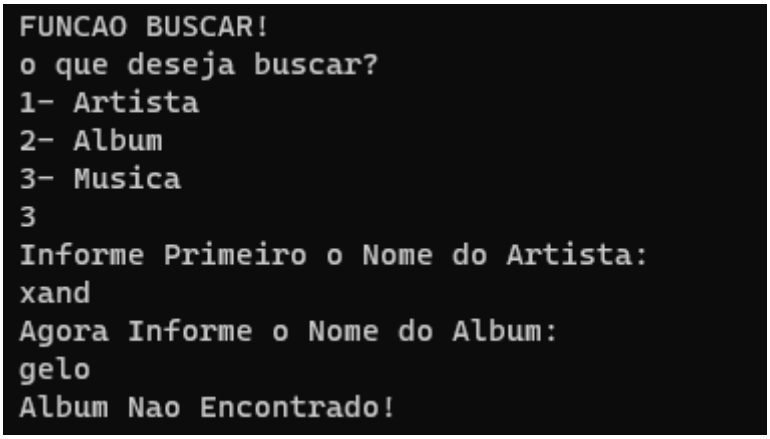

Figura 07.2

A primeira questão solicitava a realização de um experimento envolvendo a busca por 30 itens, especificamente artistas, na árvore. O objetivo era monitorar e registrar o tempo gasto durante a busca de cada item na árvore. Após a conclusão do experimento, foi realizada uma análise detalhada dos resultados obtidos. Observe a figura 8.

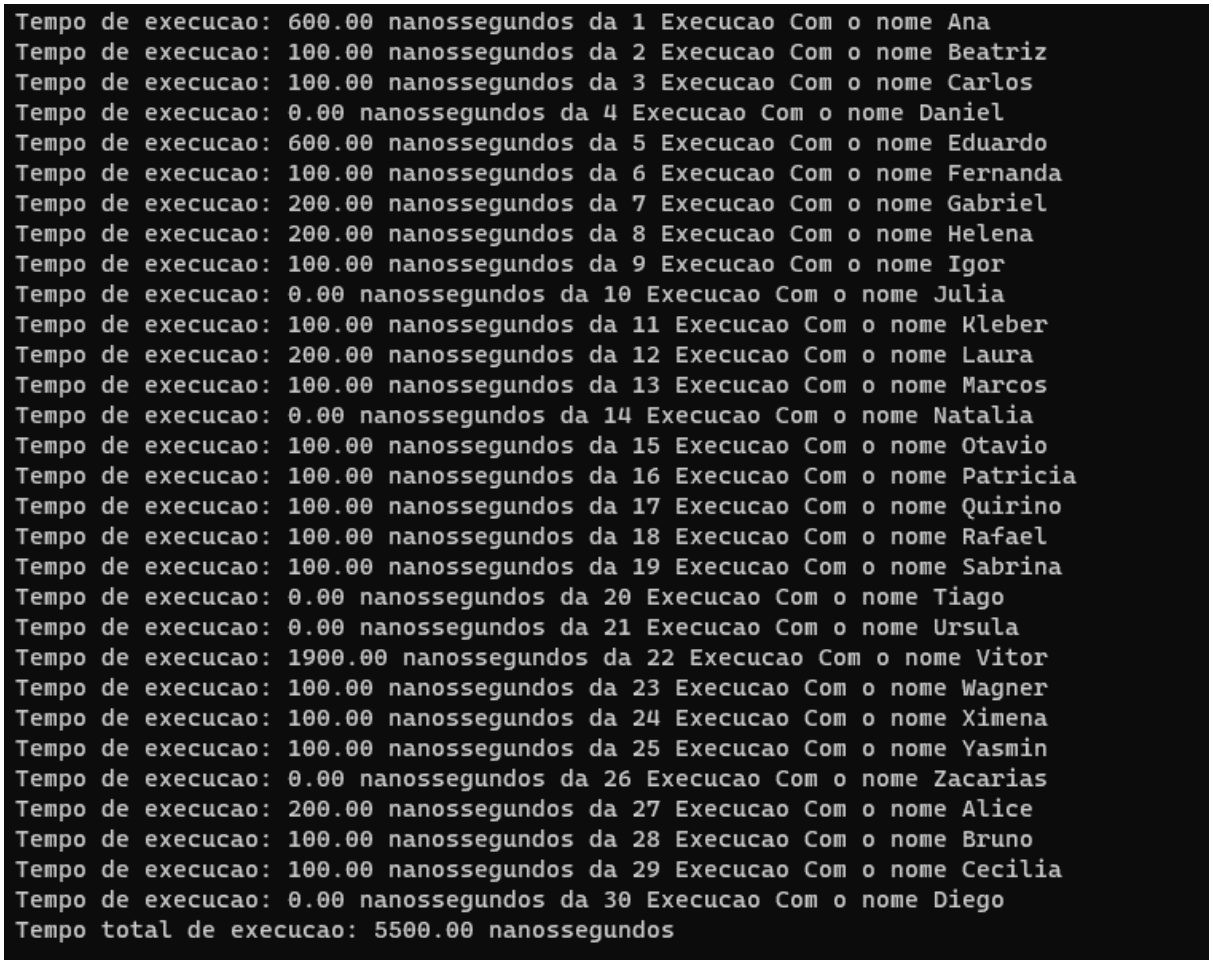

Figura 8

A segunda questão tem a mesma finalidade que a primeira, ambas visam o mesmo objetivo. A única diferença é que a segunda questão requer que a implementação siga a estrutura de dados 2-3. Ou seja, um nó contém duas informações e pode ter até três filhos. A árvore está realizando o cadastro corretamente, inserindo apenas nas folhas e mantendo a menor das duas informações à esquerda, conforme é a norma da árvore 2-3.

Ainda na questão 2 foi solicitado a realização de um experimento envolvendo a busca por 30 itens, especificamente artistas, na árvore. O objetivo era monitorar e registrar o tempo gasto durante a busca de cada item na árvore. Após a conclusão do experimento, foi realizada uma análise detalhada dos resultados obtidos. A Figura 9 apresenta o teste realizado com uma lista de 30 nomes.

```
Tempo de execucao: 300.00 nanossegundos da 1 Execucao Com o nome Ana
Tempo de execucao: 200.00 nanossegundos da 2 Execucao Com o nome Beatriz
Tempo de execucao: 200.00 nanossegundos da 3 Execucao Com o nome Carlos
Tempo de execucao: 0.00 nanossegundos da 4 Execucao Com o nome Daniel
Tempo de execucao: 600.00 nanossegundos da 5 Execucao Com o nome Eduardo
Tempo de execucao: 200.00 nanossegundos da 6 Execucao Com o nome Fernanda
Tempo de execucao: 100.00 nanossegundos da 7 Execucao Com o nome Gabriel
Tempo de execucao: 100.00 nanossegundos da 8 Execucao Com o nome Helena
Tempo de execucao: 100.00 nanossegundos da 9 Execucao Com o nome Igor
Tempo de execucao: 100.00 nanossegundos da 10 Execucao Com o nome Julia
Tempo de execucao: 100.00 nanossegundos da 11 Execucao Com o nome Kleber
Tempo de execucao: 200.00 nanossegundos da 12 Execucao Com o nome Laura
Tempo de execucao: 100.00 nanossegundos da 13 Execucao Com o nome Marcos
Tempo de execucao: 100.00 nanossegundos da 14 Execucao Com o nome Natalia
Tempo de execucao: 100.00 nanossegundos da 15 Execucao Com o nome Otavio
Tempo de execucao: 100.00 nanossegundos da 16 Execucao Com o nome Patricia
Tempo de execucao: 100.00 nanossegundos da 17 Execucao Com o nome Quirino
Tempo de execucao: 100.00 nanossegundos da 18 Execucao Com o nome Rafael
Tempo de execucao: 200.00 nanossegundos da 19 Execucao Com o nome Sabrina
Tempo de execucao: 100.00 nanossegundos da 20 Execucao Com o nome Tiago
Tempo de execucao: 100.00 nanossegundos da 21 Execucao Com o nome Ursula
Tempo de execucao: 500.00 nanossegundos da 22 Execucao Com o nome Vitor
Tempo de execucao: 100.00 nanossegundos da 23 Execucao Com o nome Wagner
Tempo de execucao: 100.00 nanossegundos da 24 Execucao Com o nome Ximena
Tempo de execucao: 100.00 nanossegundos da 25 Execucao Com o nome Yasmin
Tempo de execucao: 100.00 nanossegundos da 26 Execucao Com o nome Zacarias
Tempo de execucao: 100.00 nanossegundos da 27 Execucao Com o nome Alice
Tempo de execucao: 100.00 nanossegundos da 28 Execucao Com o nome Bruno
Tempo de execucao: 100.00 nanossegundos da 29 Execucao Com o nome Cecilia
Tempo de execucao: 100.00 nanossegundos da 30 Execucao Com o nome Diego
Tempo total de execucao: 4500.00 nanossegundos
```

Figura 9

Conclusão

Este trabalho destacou uma implementação eficaz de estrutura de dados, como Árvore binária rubro-negra, Árvore 2-3 e Árvore 4-5, na resolução de problemas complexos relacionados ao cadastramento e manipulações de arvores de artista e álbuns. A utilização dessas estruturas de dados não só proporcionou uma organização sistemática e acessível dos dados, mas também otimizou a execução de operações de inserção e buscar, reduzindo significativamente a complexidade computacional dessas operações.

A terceira questão provou ser muito complexa e, infelizmente, não pude concluí-la no tempo esperado. A primeira e segunda questão exigiu bastante esforço e consumiu mais tempo do que o previsto para ser concluída.

Os resultados obtidos através dos experimentos realizados foram bastante promissores. Foi possível desenvolver programas que os abordam os problemas proposto de maneira eficiente, lidando com a criação, manipulação e análise de arvores de artistas e álbuns de forma precisa e eficaz. Estes programas não só resolveram os problemas propostos, mas também demonstraram a aplicabilidade pratica das estruturas de dados estudadas. Além disso, este trabalho permitiu um aprofundamento no entendimento das estruturas de dados utilizadas.

Em resumo, a resolução dos problemas propostos neste trabalho não só permitiu o desenvolvimento de habilidades de programação relacionadas ao cadastramento e manipulações de arvores de series e temporadas, mas também proporcionou uma compreensão aprofundada e prática das estruturas de dados estudadas. Este trabalho, portanto, representa um passo significativo no estudo e na aplicação de estruturas de dados na resolução de problemas reais.

Apêndice

Nesta seção será mostrado os códigos fontes de cada questão realizada.

Problema 01:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
//erikfazendoteste
//cris fazendo teste


#define Black 0
#define Red 1


typedef struct Musica{
   char tituloMusica[200];
   float minutos;
   struct Musica *Prox;
}Musica;
```

```
typedef struct Albuns{
    char tituloAlbum[200];
    int ano;
    int quantmusica;
    int cor;
    struct Musica *musica;
    struct Albuns *Esq;
    struct Albuns *Dir;


}Albuns;

typedef struct Artista{
    char artista[200];
    char TipoArtista[200];
    char estilomusical[100];
    int numAlbuns;
    int Cor;
    struct Albuns *albuns;
    struct Artista *Esq;
    struct Artista *Dir;

}Artista;
```

```
int CadastraMusica(Musica **lista, char *TituloMusica, float min){
    int sabe = 1;

    Musica *aux = *lista;
    while (aux != NULL && sabe != 0) {
        if (strcmp(TituloMusica, aux->tituloMusica) == 0) {
            sabe = 0;

        }
        aux = aux->Prox;
    }
```

```
   if(sabe != 0){

      Musica *novaMusica = (Musica*)malloc(sizeof(Musica));

      strcpy(novaMusica->tituloMusica, TituloMusica);
      novaMusica->minutos = min;

      novaMusica->Prox = NULL;

      if (*lista == NULL) {
         *lista = novaMusica;
      }
      else {
         Musica *aux = *lista;
         Musica *ant = NULL;

         while (aux != NULL && strcmp(novaMusica->tituloMusica, aux-
>tituloMusica) > 0) {
            ant = aux;
            aux = aux->Prox;
         }

         if (ant == NULL) {
            novaMusica->Prox = *lista;
            *lista = novaMusica;
         }
         else {
            ant->Prox = novaMusica;
            novaMusica->Prox = aux;
         }
      }

   }

   return sabe;
}


Albuns *rotacionaDireitaAlbuns(Albuns *raiz)
{
   Albuns *Aux = raiz->Esq;
   raiz->Esq = Aux->Dir;
```

```
    Aux->Dir = raiz;
    Aux->cor = raiz->cor;
    raiz->cor = Red;

    return Aux;
}

Albuns *rotacionaEsquerdaAlbuns(Albuns *raiz)
{
    Albuns *Aux = raiz->Dir;
    raiz->Dir = Aux->Esq;
    Aux->Esq = raiz;
    Aux->cor = raiz->cor;
    raiz->cor = Red;

    return Aux;

}

void trocaCorAlbuns(Albuns *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;
    }
}

Artista *rotacionaDireita(Artista *raiz){
    Artista *Aux = (raiz)->Esq;
    (raiz)->Esq = Aux->Dir;
    Aux->Dir = raiz;
    Aux->Cor = (raiz)->Cor;
    (raiz)->Cor = Red;

    return Aux;
```

```
}

Artista  *rotacionaEsquerda(Artista *raiz) {
   Artista *aux = (raiz)->Dir;
   (raiz)->Dir = aux->Esq;
   aux->Esq = raiz;
   aux->Cor = (raiz)->Cor;
   (raiz)->Cor = Red;

   return aux;
}


void trocaCor(Artista *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;
   }
}


Artista* Balancear(Artista *raiz) {
   //No vermelho é sempre filho à esquerda
   if (raiz->Dir != NULL && raiz->Dir->Cor == Red) {
      raiz = rotacionaEsquerda(raiz);
   }

   //Filho da direita e neto da esquerda são vermelhos
   if (raiz->Esq != NULL && raiz->Esq->Cor == Red && raiz->Esq->Esq->Cor
== Red) {
      raiz = rotacionaDireita(raiz);
   }

   //Dois filhos vermelhos: trocar cor!!!
   if (raiz->Esq->Cor == Red && raiz->Dir->Cor == Red) {
      trocaCor(raiz);
   }
```

```
    return raiz;
}

Albuns* Balancear_album(Albuns *raiz) {
    //No vermelho é sempre filho à esquerda
    if (raiz->Dir != NULL && raiz->Dir->cor == Red) {
        raiz = rotacionaEsquerdaAlbuns(raiz);
    }

    //Filho da direita e neto da esquerda são vermelhos
    else if (raiz->Esq != NULL && raiz->Esq->Esq != NULL && raiz->Dir !=
NULL && raiz->Dir->cor == Red && raiz->Esq->Esq->cor == Red) {
        raiz = rotacionaDireitaAlbuns(raiz);
    }

    //Dois filhos vermelhos: trocar cor!!!
    else if (raiz->Esq != NULL && raiz->Dir != NULL && raiz->Esq->cor ==
Red && raiz->Dir->cor == Red) {
        trocaCorAlbuns(raiz);
    }

    return raiz;
}




Artista* move2EsqRED(Artista *raiz) {
    trocaCor(raiz);
    if (raiz->Dir->Esq->Cor == Red) {
        raiz->Dir = rotacionaDireita(raiz->Dir);
        raiz = rotacionaEsquerda(raiz);
        trocaCor(raiz);
    }

    return raiz;
}



Artista* move2DirRED(Artista *raiz) {
    trocaCor(raiz);
    if (raiz->Esq->Esq->Cor == Red) {
```

```
        raiz = rotacionaDireita(raiz);
        trocaCor(raiz);
    }

    return raiz;
}


Albuns* move2EsqRED_album(Albuns *raiz) {
    trocaCorAlbuns(raiz);
    if (raiz->Dir->Esq != NULL && raiz->Dir->Esq->cor == Red) {
        raiz->Dir = rotacionaDireitaAlbuns((raiz->Dir));
        raiz = rotacionaEsquerdaAlbuns(raiz);
        trocaCorAlbuns(raiz);
    }

    return raiz;
}

Albuns* move2DirRED_album(Albuns *raiz) {
    trocaCorAlbuns(raiz);
    if (raiz->Dir->Esq->cor == Red) {
        raiz->Dir = rotacionaDireitaAlbuns((raiz->Dir));
        raiz = rotacionaEsquerdaAlbuns(raiz);
        trocaCorAlbuns(raiz);
    }

    return raiz;
}

Artista* removeMenor(Artista *raiz) {
    if (raiz != NULL) {
        if (raiz->Esq->Cor == Black && (raiz->Esq->Esq->Cor == Black)) {
            raiz = move2EsqRED(raiz);
        }
        raiz->Esq = removeMenor(raiz->Esq);
        raiz = Balancear(raiz);
    } else {
        free(raiz);
        raiz = NULL;
    }
    return raiz;
}
```

```
Albuns* removeMenor_album(Albuns *raiz) {
   if (raiz != NULL) {
      if (raiz->Esq != NULL) {
         if (raiz->Esq->cor == Black && (raiz->Esq->Esq == NULL || raiz->Esq-
>Esq->cor == Black)) {
            raiz = move2EsqRED_album(raiz);
         }
         raiz->Esq = removeMenor_album(raiz->Esq);
      } else {
         free(raiz);
         raiz = NULL;
      }

      if (raiz != NULL) {
         raiz = Balancear_album(raiz);
      }
   }
   return raiz;
}



Artista* procuraMenor(Artista *raizAtual){
   Artista *no1 = raizAtual;
   Artista *no2 = raizAtual->Esq;
   while(no2 != NULL){
      no1 = no2;
      no2 = no2->Esq;
   }

   return no1;
}

Albuns* procuraMenor_album(Albuns *raizAtual){
   Albuns *no1 = raizAtual;
   Albuns *no2 = raizAtual->Esq;
   while(no2 != NULL){
      no1 = no2;
      no2 = no2->Esq;
   }
```

```
   return no1;
}

int ArtistaExiste(Artista *raiz, char *nomeArtista)
{
   return (raiz != NULL) &&
          ((strcmp(raiz->artista, nomeArtista) == 0) ||
          ArtistaExiste(raiz->Esq, nomeArtista) ||
          ArtistaExiste(raiz->Dir, nomeArtista));
}


Artista *InsereNoArtista(Artista *raiz, char *novoArtista, int *resp)
{

   if (ArtistaExiste(raiz, novoArtista))
   {
      *resp = 0;
   }

   else if (raiz == NULL)
   {
      Artista *novo = (Artista *)malloc(sizeof(Artista));
      if (novo == NULL)
      {
         *resp = 0;
      }
      else
      {
         strcpy(novo->artista, novoArtista);

         printf("Informe o Estilo Musical:\n");
         fflush(stdin);
         scanf("%[^\n]", novo->estilomusical);

         printf("Informe o Tipo do Artista:\n");
         fflush(stdin);
         scanf("%[^\n]", novo->TipoArtista);

         novo->albuns = NULL;
         novo->Cor = Red;
         novo->Dir = NULL;
         novo->Esq = NULL;
```

```
            novo->numAlbuns = 0;
            *resp = 1;
            raiz = novo;
        }
    }
    else
    {
        if (strcmp(novoArtista, raiz->artista) < 0)
        {
            raiz->Esq = InsereNoArtista(raiz->Esq, novoArtista, resp);
        }
        else
        {
            raiz->Dir = InsereNoArtista(raiz->Dir, novoArtista, resp);
        }

        if (raiz->Dir != NULL && raiz->Dir->Cor == Red && (raiz->Esq == NULL || raiz->Esq->Cor == Black))
        {
          raiz = rotacionaEsquerda(raiz);
        }

        if (raiz->Esq != NULL && raiz->Esq->Cor == Red && raiz->Esq->Esq != NULL && raiz->Esq->Esq->Cor == Red)
        {
            raiz = rotacionaDireita(raiz);
        }

        if (raiz->Esq != NULL && raiz->Esq->Cor == Red && raiz->Dir != NULL && raiz->Dir->Cor == Red)
        {
            trocaCor(raiz);
        }
    }

    return raiz;
}


int InsereArvRBArtista(Artista **raiz, char *nomeArtista) {
    int resp;
    *raiz = InsereNoArtista(*raiz, nomeArtista, &resp);
```

```
   if (*raiz != NULL) {
      (*raiz)->Cor = Black;
   }

   return resp;
}

Albuns *InsereArvRBAlbuns(Albuns *raiz, Artista *artista, char *novoAlbum,
int *resp)
{
   Albuns *retorno = raiz;


   if (artista != NULL)
   {

      if (raiz == NULL)
      {
         Albuns *novo = (Albuns *)malloc(sizeof(Albuns));
         if (novo != NULL)
         {
            strcpy(novo->tituloAlbum, novoAlbum);
            printf("Informe o Ano do Album:\n");
            scanf("%d", &novo->ano);
            printf("Informe a Quantidade de Musicas:\n");
            scanf("%d", &novo->quantmusica);
            novo->musica = NULL;
            novo->cor = Red;
            novo->Dir = NULL;
            novo->Esq = NULL;

            *resp = 1;

            artista->numAlbuns++;

            retorno = novo;
         }
         else
         {
            *resp = 0;
         }
      }
      else
```

```
    {
        int comp = strcmp(novoAlbum, raiz->tituloAlbum);
        if (comp != 0)
        {

            if (strcmp(novoAlbum, raiz->tituloAlbum) < 0)
            {
                raiz->Esq = InsereArvRBAlbuns(raiz->Esq, artista, novoAlbum,
resp);
            }
            else
            {
                raiz->Dir = InsereArvRBAlbuns(raiz->Dir, artista, novoAlbum,
resp);
            }


            if (raiz->Dir != NULL && raiz->Dir->cor == Red && (raiz->Esq ==
NULL || raiz->Esq->cor == Black))
            {
                raiz = rotacionaEsquerdaAlbuns(raiz);
            }

            if (raiz->Esq != NULL && raiz->Esq->cor == Red && raiz->Esq-
>Esq != NULL && raiz->Esq->Esq->cor == Red)
            {
                raiz = rotacionaDireitaAlbuns(raiz);
            }

            if (raiz->Esq != NULL && raiz->Esq->cor == Red && raiz->Dir !=
NULL && raiz->Dir->cor == Red)
            {
                trocaCorAlbuns(raiz);
            }
        }
        else
        {
            *resp = 0;
        }
    }
  }
  else
  {
```

```
        *resp = 0;
    }

    return retorno;
}

int InsereArvNoAlbum(Albuns **raiz, Artista *artista, char *nomeAlbum)
{
    int resp;
    *raiz = InsereArvRBAlbuns(*raiz, artista, nomeAlbum, &resp);

    if (*raiz != NULL)
    {
        (*raiz)->cor = Black;
    }

    return resp;
}


Artista *remove_NO(Artista *raiz, char *nomeArtista){
    printf("ENTROU MESMOOOO\n");
    if(strcmp(nomeArtista, raiz->artista) < 0){
        printf("ENTROU QUANDO O VALOR E MENOR\n");
        if((raiz->Esq->Cor == Black) && (raiz->Esq->Esq->Cor == Black)){
            raiz = move2EsqRED(raiz);
        }

        raiz->Esq = remove_NO(raiz->Esq, nomeArtista);
    }
    else{
        printf("Entrou no ELSE\n");
        if(raiz->Esq->Cor == Red){
            printf("ENTROU NO IF DA ROTACAO QUANDO E VERMELHO");
            raiz = rotacionaDireita(raiz);
        }

        if((strcmp(nomeArtista, raiz->artista) == 0) && (raiz->Dir == NULL)){
            printf("ENTROU QUANDO A DIR E NULL E E O VALOR QUE
QUEREMOS REMOVER\n");
            free(raiz);
            return NULL;
        }
```

```
        printf("ESTAMOS EM CIMA DA ONDE E PRA ENTRAR\n");
        printf("COR DA DIR DA RAIZ: %d\n", raiz->Dir->Cor);
        if((raiz->Dir->Cor == Black) && ((raiz->Dir->Esq->Cor == Black) || (raiz->Dir->Esq == NULL))){
            printf("ENTROU NO MOVE2DIRRED!!\n");
            raiz = move2DirRED(raiz);
        }
        printf("Passou da condicao que e pra entrar\n");
        if(strcmp(nomeArtista, raiz->artista) == 0){
            printf("ENTROU QUANDO E SO IGUAL\n");
            Artista *Aux = procuraMenor(raiz->Dir);
            strcpy(raiz->artista, Aux->artista);
            raiz->Dir = removeMenor(raiz->Dir);
        }else{
            raiz->Dir = remove_NO(raiz->Dir, nomeArtista);
        }


    }

    return Balancear(raiz);
}

Albuns *remove_NO_album(Albuns *raiz, char *nomeAlbum){
    if (raiz == NULL) {
        return NULL;
    }

    if (strcmp(nomeAlbum, raiz->tituloAlbum) < 0){
        if (raiz->Esq != NULL) {
            if (raiz->Esq->cor == Black && (raiz->Esq->Esq == NULL || raiz->Esq->Esq->cor == Black)) {
                raiz = move2EsqRED_album(raiz);
            }
            raiz->Esq = remove_NO_album(raiz->Esq, nomeAlbum);
        }
    }

    else if (strcmp(nomeAlbum, raiz->tituloAlbum) > 0){
        if (raiz->Dir != NULL) {
            if (raiz->Esq->cor == Red) {
                raiz = remove_NO_album(raiz->Dir, nomeAlbum);
```

```
        }

        if (strcmp(nomeAlbum, raiz->tituloAlbum) == 0 && (raiz->Dir ==
NULL)) {
            free(raiz);
            return NULL;
        }

        if (raiz->Dir != NULL && raiz->Dir->cor == Black && (raiz->Dir->Esq
== NULL || raiz->Dir->Esq->cor == Black)) {
            raiz = move2DirRED_album(raiz);
        }

        if (strcmp(nomeAlbum, raiz->tituloAlbum) == 0) {
            Albuns *x = procuraMenor_album(raiz->Dir);
            strcpy(raiz->tituloAlbum, x->tituloAlbum);
            raiz->Dir = removeMenor_album(raiz->Dir);
        }
        else {
            raiz->Dir = remove_NO_album(raiz->Dir, nomeAlbum);
        }
      }
   }

   raiz = Balancear_album(raiz);
   return raiz;
}


int remove_Artista(Artista **raiz, char *nomeArtista){
  int sabe;
  int resultado = 0;
  sabe = ArtistaExiste(*raiz, nomeArtista);
  if(sabe == 1){
   printf("Artista encontrado!\n");
   Artista *Aux = *raiz;
   printf("Entrando em remove no\n");
   *raiz = remove_NO(Aux, nomeArtista);
   printf("saiu de remove no\n");
   if(*raiz != NULL){
      (*raiz)->Cor = Black;
      resultado = 1;
   }
```

```
    }
    return resultado;
}


int remove_Album(Albuns **raiz, char *nomeAlbum){
    int resultado = 0;  // Não encontrou o album para remover
    if (*raiz != NULL) {
        *raiz = remove_NO_album(*raiz, nomeAlbum);
        if (*raiz != NULL) {
            (*raiz)->cor = Black;
            resultado = 1;  // Remoção bem-sucedida
        }
    }
    return resultado;
}


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

    if (strcmp(nome, raiz->artista) < 0)
        raiz->Esq = excluir_artista(raiz->Esq, nome);
    else if (strcmp(nome, raiz->artista) > 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

            Artista *temp = (raiz->Esq != NULL) ? raiz->Esq : raiz->Dir;
            free(raiz);
            return temp;
```

```
        }
        else
        { //dois filhos
            Artista *temp = procuraMenor(raiz->Dir);
            raiz = temp;
            raiz->Dir = excluir_artista(raiz->Dir, temp->artista);
        }


    }

    Balancear(raiz);

    return raiz;
}

Albuns *excluir_Album(Albuns *raiz, char *nome)
{
    if (raiz == NULL) {
        return NULL;
    }

    if (strcmp(nome, raiz->tituloAlbum) < 0)
        raiz->Esq = excluir_Album(raiz->Esq, nome);
    else if (strcmp(nome, raiz->tituloAlbum) > 0)
        raiz->Dir = excluir_Album(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

            Albuns *temp = (raiz->Esq != NULL) ? raiz->Esq : raiz->Dir;
            free(raiz);
            return temp;
        }
        else
```

```
        { //dois filhos
            Albuns *temp = procuraMenor_album(raiz->Dir);
            raiz = temp;
            raiz->Dir = excluir_Album(raiz->Dir, temp->tituloAlbum);
        }

    }

    Balancear_album(raiz);

    return raiz;
}

void  remover_musica(Musica **lista, char *TituloMusica){
    Musica *aux = *lista;
    Musica *ant = NULL;


    while (aux != NULL && strcmp(TituloMusica, aux->tituloMusica) != 0) {
        ant = aux;
        aux = aux->Prox;
    }

    if (aux != NULL) {

        if (ant == NULL) {
            *lista = aux->Prox;
        }

        else {
            ant->Prox = aux->Prox;
        }

        printf("Música '%s' removida com sucesso.\n", TituloMusica);

        free(aux);
    } else {
        printf("Música '%s' não encontrada.\n", TituloMusica);
    }


}
```

```
void imprime(Artista *raiz)
{

   if(raiz != NULL){
      imprime(raiz->Esq);
      printf("Nome do Artista: %s\n", raiz->artista);
      printf("Estilo Musical: %s\n", raiz->estilomusical);
      printf("Tipo do Artista: %s\n", raiz->TipoArtista);
      printf("Numero de Albuns: %d\n", raiz->numAlbuns);
      imprime(raiz->Dir);
   }
}

void mostrarDadosArtista(Artista *raiz, char *nomeArtista){
   if(raiz != NULL){
      if(strcmp(raiz->artista, nomeArtista) == 0){
         printf("Nome do Artista: %s\n", raiz->artista);
         printf("Estilo Musical: %s\n", raiz->estilomusical);
         printf("Quantidade de Albuns: %d\n", raiz->numAlbuns);
      }
      mostrarDadosArtista(raiz->Esq, nomeArtista);
      mostrarDadosArtista(raiz->Dir, nomeArtista);
   }
}

Artista *BuscarArtista(Artista **raiz, char *nomeArtista){
   Artista *result = NULL;
   if (*raiz!=NULL){
      if (strcmp(nomeArtista, (*raiz)->artista) == 0){
         result = *raiz;
      }
      else{
         if (strcmp(nomeArtista, (*raiz)->artista) < 0){
            result = BuscarArtista(&(*raiz)->Esq, nomeArtista);
         }
         else{
            result = BuscarArtista(&(*raiz)->Dir, nomeArtista);
         }
      }
```

```
    }

    return result;
}


Albuns *BuscarAlbum(Albuns **raiz, char *nomealbum){
    Albuns *result = NULL;
    if (*raiz!=NULL){
        if (strcmp(nomealbum, (*raiz)->tituloAlbum) == 0){
            result = *raiz;
        }
        else{
            if (strcmp(nomealbum, (*raiz)->tituloAlbum) < 0){
                result = BuscarAlbum(&(*raiz)->Esq, nomealbum);
            }
            else{
                result = BuscarAlbum(&(*raiz)->Dir, nomealbum);
            }
        }
    }

    return result;
}


Musica* BuscaMusica(Musica *lista, char *TituloMusica) {
    Musica *aux = lista;

    // Percorra a lista
    while (aux != NULL) {
        // Se o título da música atual corresponder ao título da música procurada
        if (strcmp(TituloMusica, aux->tituloMusica) == 0) {
            return aux; // Retorne a música
        }
        aux = aux->Prox;
    }

    // Se a música não foi encontrada na lista
    return NULL;
}

void imprimeAlbuns(Albuns *raiz)
```

```
{
   if (raiz != NULL)
   {
      imprimeAlbuns(raiz->Esq);
      printf("Nome do Album: %s\n", raiz->tituloAlbum);
      printf("Ano: %d\n", raiz->ano);
      printf("Quantidade de Musicas: %d\n", raiz->quantmusica);
      printf("\n");
      imprimeAlbuns(raiz->Dir);
   }
}

void imprimeMusica(Musica *lista)
{
   Musica *aux = lista;


   while (aux != NULL)
   {
      printf("Titulo da Musica: %s\n", aux->tituloMusica);
      printf("Duracao: %.2f\n", aux->minutos);
      printf("\n");
      aux = aux->Prox;
   }
}



void liberaMusica(Musica **lista){
   if(*lista != NULL){
      liberaMusica(&((*lista)->Prox));
      free(*lista);
      *lista = NULL;
   }
}

void liberaAlbum(Albuns **raiz){
   if(*raiz != NULL){
      liberaAlbum(&((*raiz)->Esq));
      liberaAlbum(&((*raiz)->Dir));
      liberaMusica(&((*raiz)->musica));
      free(*raiz);
      *raiz = NULL;
```

```
    }
}

void liberaArtista(Artista **raiz){
    if(*raiz != NULL){
        liberaArtista(&((*raiz)->Esq));
        liberaArtista(&((*raiz)->Dir));
        liberaAlbum(&((*raiz)->albuns));
        liberaMusica(&((*raiz)->albuns->musica));
        free(*raiz);
        *raiz = NULL;
    }
}


int main(){
    Artista *RaizArtista;
    RaizArtista = NULL;
    char nomeArtista[200];
    char nomeAlbum[200];
    int op;
    while(1){

        printf("==========MENU==========\n");
        printf("1- Cadastrar Artista\n");
        printf("2- Cadastrar Album\n");
        printf("3- Cadastrar Musica\n");
        printf("4- Buscar\n");
        printf("5- Remover\n");
        printf("6- Imprimir\n");
        printf("7- Sair\n");
        scanf("%d", &op);


        switch (op)
        {
        case 1:{


            int recebeu;
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);
```

```
            recebeu = InsereArvRBArtista(&RaizArtista, nomeArtista);



            if(recebeu == 1){
               printf("Artista Cadastrado Com Sucesso!\n");
            }
            else{
               printf("Artista informado ja esta cadastrado!\n");
            }
            break;

        }
        case 2:{

            int recebeu;
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);

            Artista *verifica;
            verifica = BuscarArtista(&RaizArtista, nomeArtista);

            if(verifica == NULL){
               printf("Artista nao encontrado\n");
            }
            else{

               printf("Informe o Nome do Album:\n");
               fflush(stdin);
               scanf("%[^\n]", nomeAlbum);

               recebeu = InsereArvNoAlbum(&verifica->albuns, verifica,
nomeAlbum);

               if(recebeu == 1){
                  printf("Album Cadastrado Com Sucesso!\n");
               }
               else{
                  printf("Album informado ja esta cadastrado!\n");
               }
            }
```

```
            break;
        }
        case 3:{

            printf("Nova Musica Chegando...\n");
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);

            Artista *sabeArtista;
            sabeArtista = BuscarArtista(&RaizArtista, nomeArtista);

            if(sabeArtista == NULL){
                printf("Artista Nao Encontrado!\n");
            }
            else{
                printf("Informe o Nome do Album:\n");
                fflush(stdin);
                scanf("%[^\n]", nomeAlbum);

                Albuns *sabeAlbum;
                sabeAlbum = BuscarAlbum(&sabeArtista->albuns, nomeAlbum);

                if(sabeAlbum == NULL){
                    printf("O Album Informado Nao Existe\n");
                }

                else{
                    char tituloMusica[200];
                    float minutos;
                    int ver;

                    printf("Informe o Titulo da Musica:\n");
                    fflush(stdin);
                    scanf("%[^\n]", tituloMusica);

                    printf("Informe a Quantidade de Minutos da Musica %s\n",
tituloMusica);
                    scanf("%f", &minutos);
```

```
            ver = CadastraMusica(&(sabeAlbum->musica), tituloMusica, minutos);

            if(ver == 0){
               printf("A Musica %s Ja Existe no Album %s\n", tituloMusica, nomeAlbum);

            }
            else{
               printf("A Musica %s Foi Cadastrada com Sucesso!\n", tituloMusica);

            }


         }


      }

      break;

   }

   case 4:{
      printf("FUNCAO BUSCAR!\n");
      printf("o que deseja buscar?\n");
      printf("1- Artista\n");
      printf("2- Album\n");
      printf("3- Musica\n");

      int op2;

      scanf("%d", &op2);
      if(op2 == 1){

         printf("Informe o Nome do Artista:\n");
         fflush(stdin);
         scanf("%[^\n]", nomeArtista);

        Artista *verifica;
        verifica = BuscarArtista(&RaizArtista, nomeArtista);
```

```
            if (verifica == NULL){
             printf("Artista nao encontrado\n");
            }
            else{
             mostrarDadosArtista(RaizArtista, nomeArtista);
            }
          }
          else if(op2 == 2){
             printf("Informe o Nome do Artista:\n");
             fflush(stdin);
             scanf("%[^\n]", nomeArtista);

             Artista *verifica;
             verifica = BuscarArtista(&RaizArtista, nomeArtista);

             if(verifica == NULL){
                printf("Artista nao encontrado\n");
             }
             else{
                printf("Informe o Nome do Album:\n");
                fflush(stdin);
                scanf("%[^\n]", nomeAlbum);

                Albuns *sabeAlbum;
                sabeAlbum = BuscarAlbum(&verifica->albuns, nomeAlbum);

                if(sabeAlbum == NULL){
                   printf("Album nao encontrado\n");
                }
                else{
                   printf("INFORMACOES DO ALBUM:\n");
                   printf("Nome: %s\n", sabeAlbum->tituloAlbum);
                   printf("Ano: %d\n", sabeAlbum->ano);
                   printf("Quantidade de Musicas: %d\n", sabeAlbum-
>quantmusica);
                   printf("Artista(s): %s\n", verifica->artista);
                   printf("\n");
                }
             }
          }
          else if(op2 == 3){

             Artista *arti;
```

```
            printf("Informe Primeiro o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);
            arti = BuscarArtista(&RaizArtista, nomeArtista);

            if(arti == NULL){
               printf("Artista Nao Encontrado!\n");

            }

            else{
               printf("Agora Informe o Nome do Album:\n");
               fflush(stdin);
               scanf("%[^\n]", nomeAlbum);

               Albuns *sabeAlbum;
               sabeAlbum = BuscarAlbum(&arti->albuns, nomeAlbum);

               if(sabeAlbum == NULL){
                  printf("Album Nao Encontrado!\n");
               }

               else{
                  char musica[200];
                  Musica *aux;

                  printf("Informe o Nome da Musica:\n");
                  fflush(stdin);
                  scanf("%[^\n]", musica);
                  aux = BuscaMusica(sabeAlbum->musica, musica);

                  if(aux == NULL){
                     printf("Musica Nao Encontrada\n");
                  }
                  else{

                     printf("INFORMACOES DA MUSICA:\n");
                     printf("Nome: %s\n", aux->tituloMusica);
                     printf("Minutos: %.2f\n", aux->minutos);
                     printf("Album que ela se Encontra: %s\n", sabeAlbum-
>tituloAlbum);
                     printf("Artista(s): %s\n", arti->artista);
                     printf("\n");
```

```
                }

            }
        }

    }
    break;

}

case 5:{
    printf("FUNCAO REMOVER!\n");
    printf("o que deseja remover?\n");
    printf("1- Artista\n");
    printf("2- Album\n");
    printf("3- Musica\n");

    int op3;

    scanf("%d", &op3);
    if(op3 == 1){
        char nome[200];
        printf("Informe o Nome do Artista:\n");
        fflush(stdin);
        scanf("%[^\n]", nome);
        if(ArtistaExiste(RaizArtista, nome)){
            printf("Se voce remover o artista, todos os seus albuns e musicas serao removidos tambem\n");
            printf("Deseja Continuar?\n");
            printf("1- Sim\n");
            printf("2- Nao\n");
            int op5;
            scanf("%d", &op5);
            if(op5 == 1){
                int sabe = remove_Artista(&RaizArtista, nome);
                if(sabe==1){
                    printf("Artista removido com sucesso!\n");
                }
                else{
                    printf("Nao deu para remover o Artista\n");
                }

            }else{
```

```
            printf("operacao cancelada\n");
        }

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


}
else if(op3 == 3){
    char nomeArti[200], nomealbum[200];
    printf("Informe o Nome do Artista:\n");
    fflush(stdin);
    scanf("%[^\n]", nomeArti);
    Artista *Arti;
    Arti = BuscarArtista(&RaizArtista, nomeArti);

    if(Arti == NULL){
        printf("Artista Nao Encontrado\n");
    }
    else{
        printf("Artista Encotrado!\n");
        printf("Informe o Nome do Album:\n");
        fflush(stdin);
        scanf("%[^\n]", nomealbum);
        Albuns *RecebeAlbum;
        RecebeAlbum = BuscarAlbum(&(Arti->albuns), nomealbum);

        if(RecebeAlbum == NULL){
            printf("Album Nao Encotrado\n");

        }
        else{
            char nomemusic[200];
            printf("Album Encontrado\n");
            printf("Informe o Nome da Musica:\n");
            fflush(stdin);
            scanf("%[^\n]", nomemusic);

            remover_musica(&(RecebeAlbum->musica), nomemusic);


        }
```

```
        }
    }



    break;
}


case 6:{

    printf("FUNCAO IMPRIMIR!\n");
    printf("qual informacao da arvore deseja imprimir: \n");
    printf("1- Artista\n");
    printf("2- Album\n");
    printf("3- Musica\n");

    int op4;
    scanf("%d", &op4);
    if(op4 == 1){
        printf("artistas cadastrados na arvore: \n");
        imprime(RaizArtista);
    }
    else if(op4 == 2){
        printf("Informe o Nome do Artista:\n");
        fflush(stdin);
        scanf("%[^\n]", nomeArtista);

        Artista *verifica;
        verifica = BuscarArtista(&RaizArtista, nomeArtista);

        if (verifica == NULL)
        {
            printf("Artista nao encontrado\n");
        }
        else
        {
            printf("albuns cadastrados na arvore: \n");
            imprimeAlbuns(verifica->albuns);
        }
    }
    else if(op4 == 3){
        printf("Informe o Nome do Artista:\n");
```

```
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);

            Artista *verifica;
            verifica = BuscarArtista(&RaizArtista, nomeArtista);

            if (verifica == NULL)
            {
               printf("Artista nao encontrado\n");
            }
            else
            {
               printf("Informe o Nome do Album:\n");
               fflush(stdin);
               scanf("%[^\n]", nomeAlbum);

               Albuns *sabeAlbum;
               sabeAlbum = BuscarAlbum(&verifica->albuns, nomeAlbum);

               if (sabeAlbum == NULL)
               {
                  printf("Album nao encontrado\n");
               }
               else
               {
                  printf("musicas cadastradas na arvore: \n");
                  imprimeMusica(sabeAlbum->musica);
               }
            }
         }
         else{
            printf("Opcao Invalida\n");
         }

         break;
      }
      if(op == 7){
         break;
      }
      }

}
```

```
  liberaArtista(&RaizArtista);
  return 0;



}
```

Problema 02:

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


typedef struct Musica
{
    char tituloMusica[200];
    float minutos;
    struct Musica *Prox;
} Musica;



typedef struct Albuns
{
    struct Albuns *esq;
    struct Albuns *cen;
    struct Albuns *dir;
    //DADOS DO INFO1
    char tituloAlbum1[200];
    int ano1;
    int quantmusica1;
    struct Musica *musica1;
    //DADOS DO INFO 2
    char tituloAlbum2[200];
    int ano2;
    int quantmusica2;
    struct Musica *musica2;
    int NumdeInfos;

} Albuns;
```

```
//titulo, ano, quantimusica

typedef struct Artista
{
   struct Artista *esq;
   struct Artista *cen;
   struct Artista *dir;
   //DADOS DO INFO1
   char TipoArtista1[200];
   char estilomusical1[100];
   char artista1[200];
   int numAlbuns1;
   struct Albuns *albuns1;

   //DADOS DO INFO2
   char TipoArtista2[200];
   char estilomusical2[100];
   char artista2[200];
   int numAlbuns2;
   struct Albuns *albuns2;

   int NumdeInfos;

} Artista;

int ehFolha(Artista *Raiz)
{
   return (Raiz->cen == NULL && Raiz->esq == NULL && Raiz->dir ==
NULL);
}

Artista *criaNo(char *artista1, char *TipoArtista1, char *estilomusical1, int
numAlbuns1, Artista *Esq, Artista *Cen, Artista *Dir)
{
   Artista *No;

   No = (struct Artista *)malloc(sizeof(struct Artista));

   strcpy((*No).TipoArtista1, TipoArtista1);
   strcpy((*No).estilomusical1, estilomusical1);
   strcpy((*No).artista1, artista1);
   (*No).numAlbuns1 = numAlbuns1;
```

```
    (*No).albuns1 = NULL;


    strcpy((*No).artista2, " ");
    (*No).NumdeInfos = 1;
    (*No).esq = Esq;
    (*No).cen = Cen;
    (*No).dir = Dir;

    return No;
}



void adiciona(Artista **Raiz, char *nomeArtista, char *TipoArtista, char
*EstiloMus, int numdeAlbuns, Albuns *Album, Artista *MaiorNo)
{
    if (strcmp(nomeArtista, (*Raiz)->artista1) > 0)
    {
        strcpy((*Raiz)->artista2, nomeArtista);
        strcpy((*Raiz)->TipoArtista2, TipoArtista);
        strcpy((*Raiz)->estilomusical2, EstiloMus);
        (*Raiz)->albuns2 = Album;
        (*Raiz)->numAlbuns2 = numdeAlbuns;
        (*Raiz)->dir = MaiorNo;
    }
    else
    {
        strcpy((*Raiz)->artista2, (*Raiz)->artista1);
        strcpy((*Raiz)->TipoArtista2, (*Raiz)->TipoArtista1);
        strcpy((*Raiz)->estilomusical2, (*Raiz)->estilomusical1);
        (*Raiz)->numAlbuns2 = (*Raiz)->numAlbuns1;
        (*Raiz)->albuns2 = (*Raiz)->albuns1;


        strcpy((*Raiz)->artista1, nomeArtista);
        strcpy((*Raiz)->TipoArtista1, TipoArtista);
        strcpy((*Raiz)->estilomusical1, EstiloMus);
        (*Raiz)->numAlbuns1 = numdeAlbuns;
        (*Raiz)->albuns1 = Album;
        (*Raiz)->dir = (*Raiz)->cen;
        (*Raiz)->cen = MaiorNo;
    }
```

```
   (*Raiz)->NumdeInfos = 2;
}


 Artista *quebraNo(Artista **Raiz, char *artista, char *TipoArtista, char *estilo,
int numAlbuns, Albuns *Album, Albuns **SobeAlbum, char *sobeArtista, char
*sobetipoArtista, char *sobeEstilo, int *sobeNumAlbuns, Artista *MaiorNo)
{
   Artista *Novo;
   if (strcmp(artista,(*Raiz)->artista1) < 0)
   {
      // *sobe = (*Raiz)->Info1;
      strcpy(sobeArtista,(*Raiz)->artista1);
      strcpy(sobeEstilo,(*Raiz)->estilomusical1);
      strcpy(sobetipoArtista,(*Raiz)->TipoArtista1);
      *sobeNumAlbuns =(*Raiz)->numAlbuns1;
      *SobeAlbum = (*Raiz)->albuns1;

      Novo = criaNo((*Raiz)->artista2,(*Raiz)->estilomusical2,(*Raiz)-
>TipoArtista2,(*Raiz)->numAlbuns2, (*Raiz)->cen,(*Raiz)->dir, NULL);
      // (*Raiz)->Info1 = valor;
      strcpy((*Raiz)->artista2, artista);
      strcpy((*Raiz)->estilomusical2, estilo);
      strcpy((*Raiz)->TipoArtista2, TipoArtista);
      (*Raiz)->numAlbuns2 = numAlbuns;
      (*Raiz)->albuns2 = Album;
      (*Raiz)->cen = MaiorNo;
   }
   else if (strcmp(artista, (*Raiz)->artista2) < 0)
   {
      strcpy(sobeArtista, artista);
      strcpy(sobeEstilo, estilo);
      strcpy(sobetipoArtista, TipoArtista);
      *sobeNumAlbuns = numAlbuns;
      *SobeAlbum = Album;
      Novo = criaNo((*Raiz)->artista2, (*Raiz)->estilomusical2, (*Raiz)-
>TipoArtista2, (*Raiz)->numAlbuns2, MaiorNo, (*Raiz)->dir, NULL);
   }
   else
   {
      strcpy(sobeArtista, (*Raiz)->artista2);
      strcpy(sobeEstilo, (*Raiz)->estilomusical2);
```

```
      strcpy(sobetipoArtista, (*Raiz)->TipoArtista2);
      *sobeNumAlbuns = (*Raiz)->numAlbuns2;
      *SobeAlbum = (*Raiz)->albuns2;
      Novo = criaNo(artista, estilo, TipoArtista, numAlbuns, (*Raiz)->dir,
MaiorNo, NULL);
   }
   strcpy((*Raiz)->artista2, " ");



   (*Raiz)->NumdeInfos = 1;
   (*Raiz)->dir = NULL;

   return (Novo);
}
```

```
Artista *inserirArv23(Artista *Pai, Artista **Raiz, char *nomeArtista, char
*TipoArtista, char *EstiloMus, int numdeAlbuns, Albuns *Album, Albuns
**SobeAlbum, char *sobenome, char *sobetipo, char *sobeestilo, int
*sobenumalbuns)
{
   Artista  *maiorNo;
   maiorNo = NULL;
   if (*Raiz == NULL)
      *Raiz = criaNo(nomeArtista, TipoArtista, EstiloMus, numdeAlbuns,
NULL, NULL, NULL);
   else
   {
      if (ehFolha(*Raiz))
      {
         if ((*Raiz)->NumdeInfos == 1)
            adiciona(Raiz, nomeArtista, TipoArtista, EstiloMus, numdeAlbuns,
Album, maiorNo);
         else // quando não tem espaço
         {
            Artista *novo;
            novo = quebraNo(Raiz, nomeArtista, TipoArtista, EstiloMus,
numdeAlbuns, Album, SobeAlbum, sobenome, sobetipo, sobeestilo,
sobenumalbuns, maiorNo);
            if (Pai == NULL)
```

```
            {
               Artista *no;
               no = criaNo(sobenome, sobetipo, sobeestilo, *sobenumalbuns,
*Raiz, novo, NULL);
               *Raiz = no;
            }
            else
               maiorNo = novo;
         }
      }
      else
      { // quando não é folha
         if (strcmp(nomeArtista, (*Raiz)->artista1) < 0)
            maiorNo = inserirArv23(*Raiz, &((*Raiz)->esq), nomeArtista,
TipoArtista, EstiloMus, numdeAlbuns, Album, SobeAlbum, sobenome, sobetipo,
sobeestilo, sobenumalbuns);
         else if ((*Raiz)->NumdeInfos == 1 || strcmp(nomeArtista, (*Raiz)-
>artista2) < 0)
            maiorNo = inserirArv23(*Raiz, &((*Raiz)->cen), nomeArtista,
TipoArtista, EstiloMus, numdeAlbuns, Album, SobeAlbum, sobenome, sobetipo,
sobeestilo, sobenumalbuns);
         else
            maiorNo = inserirArv23(*Raiz, &((*Raiz)->dir), nomeArtista,
TipoArtista, EstiloMus, numdeAlbuns, Album, SobeAlbum, sobenome, sobetipo,
sobeestilo, sobenumalbuns);

         if (maiorNo != NULL)
         {
            if ((*Raiz)->NumdeInfos == 1)
            {
               adiciona(Raiz, sobenome, sobetipo, sobeestilo, *sobenumalbuns,
*SobeAlbum,maiorNo);
               maiorNo = NULL;
            }
            else // quando não tem espaço
            {
               char *sobeartista1 = (char *)malloc(sizeof(char) * 200),
sobetipo1[200], sobeestilo1[200];
               int sobenumdealbuns1;
               Artista *novo;
               Albuns *SobeAlbum1 = NULL;
```

```
                novo = quebraNo(Raiz, sobenome, sobetipo, sobeestilo,
*sobenumalbuns, *SobeAlbum, &SobeAlbum1,sobeartista1, sobetipo1,
sobeestilo1, &sobenumdealbuns1, maiorNo);
                if (Pai == NULL)
                {
                   Artista *no;
                   no = criaNo(sobeartista1, sobetipo1, sobeestilo1,
sobenumdealbuns1, *Raiz, novo, NULL);
                   *Raiz = no;
                   maiorNo = NULL;
                }
                else
                {
                   maiorNo = novo;
                   strcpy(sobenome, sobeartista1);
                   strcpy(sobetipo, sobetipo1);
                   strcpy(sobeestilo, sobeestilo1);
                   *SobeAlbum = SobeAlbum1;
                   *sobenumalbuns = sobenumdealbuns1;
                }
             }
          }
       }
    }

    return maiorNo;
}




void ImprimirArtista(Artista *Raiz){
    if(Raiz != NULL){

        ImprimirArtista(Raiz->esq);
        printf("Nome do Artista: %s\n", Raiz->artista1);
        printf("Estilo Musical do Artista: %s\n", Raiz->estilomusical1);
        printf("Tipo do Artista: %s\n", Raiz->TipoArtista1);
        printf("Quantidade de Albuns do Artista %s: %d\n", Raiz->artista1, Raiz-
>numAlbuns1);

        ImprimirArtista(Raiz->cen);
```

```
        if(Raiz->NumdeInfos == 2){
            printf("Imprimindo o INFO2:\n");
            printf("Nome do Artista: %s\n", Raiz->artista2);
            printf("Estilo Musical do Artista: %s\n", Raiz->estilomusical2);
            printf("Tipo do Artista: %s\n", Raiz->TipoArtista2);
            printf("Quantidade de Albuns do Artista %s: %d\n", Raiz->artista2, Raiz-
>numAlbuns2);
        }



        ImprimirArtista(Raiz->dir);
    }
}

Artista *BuscaArtista(Artista **tree, char *nomeArtista)
{
    Artista *NO;
    NO = NULL;
    if (*tree != NULL)
    {
        if (strcmp(nomeArtista, (*tree)->artista1) == 0)
        {
            NO = *tree;
        }
        else if (strcmp(nomeArtista, (*tree)->artista2) == 0)
        {
            NO = *tree;
        }
        else
        {
            if (strcmp(nomeArtista, (*tree)->artista1) < 0)
            {

                NO = BuscaArtista(&(*tree)->esq, nomeArtista);
            }
            else if (strcmp(nomeArtista, (*tree)->artista2) < 0 || (*tree)->NumdeInfos
== 1)
            {

                NO = BuscaArtista(&(*tree)->cen, nomeArtista);
            }
            else
```

```
        {

            NO = BuscaArtista(&(*tree)->dir, nomeArtista);
        }
      }
   }
   return NO;
}

//CODIGO DE ALBUNS

//titulo, ano, quantimusica

int ehFolhaAlbuns(Albuns *Raiz)
{
   return (Raiz->cen == NULL && Raiz->esq == NULL && Raiz->dir == NULL);


}

Albuns *criaNoAlbum(char *titulo1, int ano1, int quantmusica1, Albuns *esq, Albuns *cen, Albuns *dir)
{
   Albuns *No;

   No = (Albuns *)malloc(sizeof(Albuns));

   strcpy((*No).tituloAlbum1, titulo1);
   (*No).ano1 = ano1;
   (*No).quantmusica1 = quantmusica1;
   (*No).musica1 = NULL;

   strcpy((*No).tituloAlbum2, " ");
   (*No).NumdeInfos = 1;
   (*No).esq = esq;
   (*No).cen = cen;
   (*No).dir = dir;

   return No;


}
```

```
void adicionaAlbum(Albuns **Raiz, char *titulo, int anoalbum, int quantmusic,
Albuns *MaiorNo)
{
   if (strcmp(titulo, (*Raiz)->tituloAlbum1) > 0)
   {
      strcpy((*Raiz)->tituloAlbum2, titulo);

      (*Raiz)->ano2 = anoalbum;
      (*Raiz)->quantmusica2 =  quantmusic;
      (*Raiz)->dir = MaiorNo;
   }
   else
   {
      strcpy((*Raiz)->tituloAlbum2, (*Raiz)->tituloAlbum1);
      (*Raiz)->ano2 =  (*Raiz)->ano1;
      (*Raiz)->quantmusica2 = (*Raiz)->quantmusica1;



      strcpy((*Raiz)->tituloAlbum1, titulo);
      (*Raiz)->ano1 = anoalbum;
      (*Raiz)->quantmusica1 = quantmusic;

      (*Raiz)->dir = (*Raiz)->cen;
      (*Raiz)->cen = MaiorNo;
   }

   (*Raiz)->NumdeInfos = 2;
}


Albuns *quebraNoAlbum(Albuns **Raiz, char *titulo, int ano, int quantmusica,
char *sobeTitulo, int *sobeAno, int *sobeQuant, Albuns *MaiorNo)
{
   Albuns *Novo;
   if (strcmp(titulo,(*Raiz)->tituloAlbum1) < 0)
   {
      // *sobe = (*Raiz)->Info1;
      strcpy(sobeTitulo,(*Raiz)->tituloAlbum1);
```

```
        //strcpy(sobeAno,(*Raiz)->ano1);
        //strcpy(sobeQuant,(*Raiz)->quantmusica1);
        *sobeAno =(*Raiz)->ano1;
        *sobeQuant =(*Raiz)->quantmusica1;


        Novo = criaNoAlbum((*Raiz)->tituloAlbum2,(*Raiz)->ano2,(*Raiz)->quantmusica2,(*Raiz)->cen,(*Raiz)->dir, NULL);
        // (*Raiz)->Info1 = valor;
        strcpy((*Raiz)->tituloAlbum2, titulo);
        //strcpy((*Raiz)->ano2, ano);
        //strcpy((*Raiz)->quantmusica2, quantmusica);
        (*Raiz)->ano2 = ano;
        (*Raiz)->quantmusica2 = quantmusica;
        (*Raiz)->cen = MaiorNo;
    }
    else if (strcmp(titulo, (*Raiz)->tituloAlbum2) < 0)
    {
        strcpy(sobeTitulo, titulo);
        //strcpy(sobeEstilo, estilo);
        //strcpy(sobetipoArtista, TipoArtista);
        *sobeAno = ano;
        *sobeQuant = quantmusica;
        Novo = criaNoAlbum((*Raiz)->tituloAlbum2, (*Raiz)->ano2, (*Raiz)->quantmusica2, MaiorNo, (*Raiz)->dir, NULL);
    }
    else
    {
        strcpy(sobeTitulo, (*Raiz)->tituloAlbum2);
        //strcpy(sobeAno, (*Raiz)->ano2);
        //strcpy(sobetipoArtista, (*Raiz)->TipoArtista2);
        *sobeAno = (*Raiz)->ano2;
        *sobeQuant = (*Raiz)->quantmusica2;
        Novo = criaNoAlbum(titulo, ano, quantmusica, (*Raiz)->dir, MaiorNo, NULL);
    }
    strcpy((*Raiz)->tituloAlbum2, " ");



    (*Raiz)->NumdeInfos = 1;
    (*Raiz)->dir = NULL;
```

```
   return (Novo);
}


Albuns *inserirArv23Album(Albuns *Pai, Albuns **Raiz, char *titulo, int ano, int quantimusic,  char *sobetitulo, int *sobeano, int *sobequantimusic)
{
   Albuns  *maiorNo;
   maiorNo = NULL;
   if (*Raiz == NULL)
      *Raiz = criaNoAlbum(titulo, ano, quantimusic, NULL, NULL, NULL);
   else
   {
      if (ehFolhaAlbuns(*Raiz))
      {
         if ((*Raiz)->NumdeInfos == 1)
            adicionaAlbum(Raiz, titulo, ano, quantimusic, maiorNo);
         else // quando não tem espaço
         {
            Albuns *novo;
            novo = quebraNoAlbum(Raiz, titulo, ano, quantimusic, sobetitulo, sobeano, sobequantimusic, maiorNo);
            if (Pai == NULL)
            {
               Albuns *no;
               no = criaNoAlbum(sobetitulo, *sobeano, *sobequantimusic, *Raiz, novo, NULL);
               *Raiz = no;
            }
            else
               maiorNo = novo;
         }
      }
      else
      { //quando não é folha
         if (strcmp(titulo, (*Raiz)->tituloAlbum1) < 0)
            maiorNo = inserirArv23Album(*Raiz, &((*Raiz)->esq), titulo, ano, quantimusic, sobetitulo, sobeano, sobequantimusic);
         else if ((*Raiz)->NumdeInfos == 1 || strcmp(titulo, (*Raiz)->tituloAlbum2) < 0)
            maiorNo = inserirArv23Album(*Raiz, &((*Raiz)->cen), titulo, ano, quantimusic, sobetitulo, sobeano, sobequantimusic);
         else
```

```
            maiorNo = inserirArv23Album(*Raiz, &((*Raiz)->dir), titulo, ano, quantimusic, sobetitulo, sobeano, sobequantimusic);

         if (maiorNo != NULL)
         {
            if ((*Raiz)->NumdeInfos == 1)
            {
               adicionaAlbum(Raiz, sobetitulo, *sobeano, *sobequantimusic, maiorNo);
               maiorNo = NULL;
            }
            else //quando não tem espaço
            {
               char *sobetitulo1 = (char *)malloc(sizeof(char) * 200);
               int sobeano1;
               int sobequantimusic1;
               Albuns *novo;
               novo = quebraNoAlbum(Raiz, sobetitulo, *sobeano, *sobequantimusic, sobetitulo1, &sobeano1, &sobequantimusic1, maiorNo);
               if (Pai == NULL)
               {
                  Albuns *no;
                  no = criaNoAlbum(sobetitulo1, sobeano1, sobequantimusic1, *Raiz, novo, NULL);
                  *Raiz = no;
                  maiorNo = NULL;
               }
               else
               {
                  maiorNo = novo;
                  strcpy(sobetitulo, sobetitulo1);
                  *sobeano = sobeano1;
                  *sobequantimusic = sobequantimusic1;

               }
            }
         }
      }

   return maiorNo;
}
```

```
void ImprimirAlbum(Albuns *Raiz){
   if(Raiz != NULL){

      ImprimirAlbum(Raiz->esq);
      printf("Titulo do Album: %s\n", Raiz->tituloAlbum1);
      printf("Ano do Album: %d\n", Raiz->ano1);
      printf("Quantidade de Musicas: %d\n", Raiz->quantmusica1);


      ImprimirAlbum(Raiz->cen);
      if(Raiz->NumdeInfos == 2){
         printf("Imprimindo o INFO2:\n");
         printf("Titulo do Album: %s\n", Raiz->tituloAlbum2);
         printf("Ano do Album: %d\n", Raiz->ano2);
         printf("Quantidade de Musicas: %d\n", Raiz->quantmusica2);
      }



      ImprimirAlbum(Raiz->dir);
   }
}


Albuns *BuscarAlbum(Albuns **raiz, char *nomeAlbum)
{
   Albuns *NO;
   NO = NULL;
   if (*raiz != NULL)
   {
      if (strcmp(nomeAlbum, (*raiz)->tituloAlbum1) == 0)
      {
         NO = *raiz;
      }
      else if (strcmp(nomeAlbum, (*raiz)->tituloAlbum2) == 0)
      {
         NO = *raiz;
      }
      else
      {
         if (strcmp(nomeAlbum, (*raiz)->tituloAlbum1) < 0)
```

```
        {

            NO = BuscarAlbum(&(*raiz)->esq, nomeAlbum);
        }
        else if (strcmp(nomeAlbum, (*raiz)->tituloAlbum2) < 0 || (*raiz)->NumdeInfos == 1)
        {

            NO = BuscarAlbum(&(*raiz)->cen, nomeAlbum);
        }
        else
        {

            NO = BuscarAlbum(&(*raiz)->dir, nomeAlbum);
        }
    }
  }
  return NO;
}


int CadastraMusica(Musica **lista, char *TituloMusica, float min){
  int sabe = 1;
  //Verificar se a música já existe na lista
  Musica *aux = *lista;
  while (aux != NULL && sabe != 0) {
    if (strcmp(TituloMusica, aux->tituloMusica) == 0) {
      sabe = 0;

    }
    aux = aux->Prox;
  }

  if(sabe != 0){
  //Se a música não estiver na lista, prossiga com o cadastro
    Musica *novaMusica = (Musica*)malloc(sizeof(Musica));

    //GUARDANDO OS DADOS
    strcpy(novaMusica->tituloMusica, TituloMusica);
    novaMusica->minutos = min;

    novaMusica->Prox = NULL;
```

```
        if (*lista == NULL) {
            *lista = novaMusica;
        }
        else {
            Musica *aux = *lista;
            Musica *ant = NULL;

            while (aux != NULL && strcmp(novaMusica->tituloMusica, aux->tituloMusica) > 0) {
                ant = aux;
                aux = aux->Prox;
            }

            if (ant == NULL) {
                novaMusica->Prox = *lista;
                *lista = novaMusica;
            }
            else {
                ant->Prox = novaMusica;
                novaMusica->Prox = aux;
            }
        }

    }

    return sabe;
}


Musica* BuscaMusica(Musica *lista, char *TituloMusica) {
    Musica *aux = lista;

    // Percorra a lista
    while (aux != NULL) {
        // Se o título da música atual corresponder ao título da música procurada
        if (strcmp(TituloMusica, aux->tituloMusica) == 0) {
            return aux; // Retorne a música
        }
        aux = aux->Prox;
    }

    // Se a música não foi encontrada na lista
    return NULL;
```

```
}

void ImprimeMusicas(Musica *lista) {
   Musica *aux = lista;
   printf("\n");

   while (aux != NULL) {
      printf("Titulo da Musica: %s\n", aux->tituloMusica);
      printf("Duracao: %.2f minutos\n", aux->minutos);
      aux = aux->Prox;
   }
}

void  remover_musica(Musica **lista, char *TituloMusica){
   Musica *aux = *lista;
   Musica *ant = NULL;


   while (aux != NULL && strcmp(TituloMusica, aux->tituloMusica) != 0) {
      ant = aux;
      aux = aux->Prox;
   }

   if (aux != NULL) {

      if (ant == NULL) {
         *lista = aux->Prox;
      }

      else {
         ant->Prox = aux->Prox;
      }

      printf("Música '%s' removida com sucesso.\n", TituloMusica);

      free(aux);
   } else {
      printf("Música '%s' não encontrada.\n", TituloMusica);
   }


}
```

```
void liberarMusica(Musica **Lista){
   if(*Lista != NULL){
      liberarMusica(&((*Lista)->Prox));
      free(*Lista);
      *Lista = NULL;
   }
}

void liberarAlbuns(Albuns **Raiz){
   if(*Raiz != NULL){
      liberarAlbuns(&((*Raiz)->esq));
      liberarAlbuns(&((*Raiz)->cen));
      liberarAlbuns(&((*Raiz)->dir));
      liberarMusica(&((*Raiz)->musica1));
      liberarMusica(&((*Raiz)->musica2));
      free(*Raiz);
      *Raiz = NULL;
   }
}

void liberarArtista(Artista **Raiz){
   if(*Raiz != NULL){
      liberarArtista(&((*Raiz)->esq));
      liberarArtista(&((*Raiz)->cen));
      liberarArtista(&((*Raiz)->dir));
      liberarAlbuns(&((*Raiz)->albuns1));
      liberarAlbuns(&((*Raiz)->albuns2));
      free(*Raiz);
      *Raiz = NULL;
   }
}


int main()
{

   Artista *raiz;
   raiz = NULL;


   int op;
```

```
while(1){
    printf("======MENU======\n");
    printf("1 - Inserir artista\n");
    printf("2 - Cadastrar Album\n");
    printf("3 - Imprimir Artista\n");
    printf("4 - Imprimir Todos os Albuns de Um Artista\n");
    printf("5 - Imprimir Musicas De Um Album\n");
    printf("6 - Buscar Artista\n");
    printf("7 - Buscar Album\n");
    printf("8 - Buscar Musica\n");
    printf("9 - Remover Musica\n");
    printf("10 - Sair\n");

    scanf("%d", &op);

    switch(op){
        case 1:{
            char nomeArtista[200], TipoArti[200], EstiloMusical[100];
            char sobenome[200], sobetipo[200], sobeestilo[200];
            Albuns *Album = NULL;
            Albuns *SobeAlbum = NULL;



            int numAlbum, sobenumalbuns;

            printf("Digite o nome do artista: ");

            scanf("%s", nomeArtista);
            printf("Digite o tipo de artista: ");
            scanf("%s", TipoArti);
            printf("Digite o estilo musical: ");
            scanf("%s", EstiloMusical);
            printf("Digite o numero de albuns: ");
            scanf("%d", &numAlbum);



            inserirArv23(NULL, &raiz, nomeArtista, TipoArti, EstiloMusical, numAlbum, Album, &SobeAlbum,sobenome, sobetipo, sobeestilo, &sobenumalbuns);
```

```
    break;
}

case 2:{
    char nomeArt[200];
    printf("Informe o Nome do Artista:\n");
    fflush(stdin);
    scanf("%[^\n]", nomeArt);

    Artista *CadAlbum;

    CadAlbum = BuscaArtista(&raiz, nomeArt);

    if(CadAlbum == NULL){
        printf("Artista Nao Encontrado\n");
    }
    else{
        char TituloAlbum[200];
        int ano, quantimusic;
        char sobetitulo[200];
        int sobeano, sobequantimusic;
        printf("Informe o Titulo do Album:\n");
        fflush(stdin);
        scanf("%[^\n]", TituloAlbum);
        Albuns *Busca1, *Busca2;
        Busca1 = BuscarAlbum(&(CadAlbum->albuns1), TituloAlbum);
        Busca2 = BuscarAlbum(&(CadAlbum->albuns2), TituloAlbum);
        if((Busca1 != NULL || Busca2 != NULL)){
            printf("Esse Album Ja Existe Para o Artista Informado\n");
        }
        else{

            printf("Informe o Ano:\n");
            scanf("%d", &ano);

            printf("Informe a Quantidade de Musica:\n");
            scanf("%d", &quantimusic);

            int quantidade = 0;
            int ver;
            char nomemusic[200];
```

```
                float minutos;

                if(strcmp(nomeArt, CadAlbum->artista1) == 0){

                   inserirArv23Album(NULL, &(CadAlbum->albuns1), TituloAlbum, ano, quantimusic, sobetitulo, &sobeano, &sobequantimusic);


                   while(quantidade < CadAlbum->albuns1->quantmusica1){
                      printf("Informe o Titulo da Musica %d:\n", quantidade+1);
                      fflush(stdin);
                      scanf("%[^\n]", nomemusic);
                      printf("Informe o Minuto da Musica %d:\n", quantidade+1);
                      scanf("%f", &minutos);

                      ver = CadastraMusica(&(CadAlbum->albuns1->musica1), nomemusic, minutos);

                       if(ver == 0){
                         printf("A Musica %s Ja Existe no Album %s\n", nomemusic, CadAlbum->albuns1->tituloAlbum1);
                         quantidade--;


                      }
                      else{
                         printf("A Musica %s Foi Cadastrada com Sucesso!\n", nomemusic);
                         quantidade++;
                      }


                   }

                }
                else if (strcmp(nomeArt, CadAlbum->artista2) == 0){
                   inserirArv23Album(NULL, &(CadAlbum->albuns2), TituloAlbum, ano, quantimusic, sobetitulo, &sobeano, &sobequantimusic);

                    while(quantidade < CadAlbum->albuns1->quantmusica1){
                      printf("Informe o Titulo da Musica %d:\n", quantidade+1);
                      fflush(stdin);
                      scanf("%[^\n]", nomemusic);
```

```
                    printf("Informe o Minuto da Musica %d:\n", quantidade+1);
                    scanf("%f", &minutos);

                    ver = CadastraMusica(&(CadAlbum->albuns1->musica1),
nomemusic, minutos);

                     if(ver == 0){
                       printf("A Musica %s Ja Existe no Album %s\n",
nomemusic, CadAlbum->albuns1->tituloAlbum1);
                       quantidade--;


                    }
                    else{
                       printf("A Musica %s Foi Cadastrada com Sucesso!\n",
nomemusic);
                       quantidade++;
                    }


                }


            }
        }



    }

    break;

}
case 3:{
    ImprimirArtista(raiz);
    break;
}

case 4:{
    char busca[200];
    printf("Informe o Nome do Artista:\n");
    fflush(stdin);
    scanf("%[^\n]", busca);
```

```
            Artista *Buscou;

            Buscou = BuscaArtista(&raiz, busca);

            if(Buscou == NULL){
               printf("Artista nao Encontrado\n");
            }

            else{
               if(strcmp(busca, Buscou->artista1) == 0){
                  if(Buscou->albuns1 == NULL){
                     printf("Nao Tem Nenhum Album Cadastrado Para Esse
Artista\n");
                  }
                  else{
                     ImprimirAlbum(Buscou->albuns1);
                  }


               }
               else if (strcmp(busca, Buscou->artista2) == 0){
                   if(Buscou->albuns2 == NULL){
                     printf("Nao Tem Nenhum Album Cadastrado Para Esse
Artista\n");
                  }
                  else{
                     ImprimirAlbum(Buscou->albuns2);
                  }
               }
            }
            break;
         }
         case 5:{
            char busca[200], buscaalbum[200];
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", busca);
            Artista *Buscou;
            Albuns *BuscouAlbum;
            Buscou = BuscaArtista(&raiz, busca);

            if(Buscou == NULL){
               printf("Artista nao Encontrado\n");
```

```
            }

            else{
               if(strcmp(busca, Buscou->artista1) == 0){
                  printf("Informe o Nome do Album:\n");
                  fflush(stdin);
                  scanf("%[^\n]", buscaalbum);

                  BuscouAlbum = BuscarAlbum(&(Buscou->albuns1),
buscaalbum);
                  if(BuscouAlbum == NULL){
                     printf("Album Nao Encotrado\n");
                  }
                  else{
                     printf("MUSICAS:\n");
                     ImprimeMusicas(BuscouAlbum->musica1);
                  }


               }
               else if (strcmp(busca, Buscou->artista2) == 0){
                  printf("Informe o Nome do Album:\n");
                  fflush(stdin);
                  scanf("%[^\n]", buscaalbum);

                  BuscouAlbum = BuscarAlbum(&(Buscou->albuns2),
buscaalbum);
                  if(BuscouAlbum == NULL){
                     printf("Album Nao Encotrado\n");
                  }
                  else{
                     printf("MUSICAS\n");
                     ImprimeMusicas(BuscouAlbum->musica2);
                  }

               }
            }
            break;

         }
         case 6:{
            char busca[200];
            printf("Informe o Nome do Artista:\n");
```

```
            scanf("%s", busca);
            Artista *Buscou;
            Buscou = BuscaArtista(&raiz, busca);

            if(Buscou == NULL){
               printf("Artista nao Encontrado\n");
            }

            else{
               if(strcmp(busca, Buscou->artista1) == 0){
                  printf("O Artista encontrado esta no INFO1\n");
                  printf("Nome do Artista: %s\n", Buscou->artista1);
                  printf("Estilo Musical: %s\n", Buscou->estilomusical1);
                  printf("Tipo do Artista: %s\n", Buscou->TipoArtista1);
                  printf("Numero de Albuns do %s: %d \n", Buscou->artista1,
Buscou->numAlbuns1);
               }
               else if (strcmp(busca, Buscou->artista2) == 0){
                  printf("O Artista encontrado esta no INFO2\n");
                  printf("Nome do Artista: %s\n", Buscou->artista2);
                  printf("Estilo Musical: %s\n", Buscou->estilomusical2);
                  printf("Tipo do Artista: %s\n", Buscou->TipoArtista2);
                  printf("Numero de Albuns do %s: %d \n", Buscou->artista2,
Buscou->numAlbuns2);
               }
            }
            break;
         }

         case 7:{
            char nomeArtista[200];
            char nomeAlbum[200];
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);

            Artista *verifica;
            verifica = BuscaArtista(&raiz, nomeArtista);

            if(verifica == NULL){
               printf("Artista nao encontrado\n");
            }
            else{
```

```
            printf("Informe o Nome do Album:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeAlbum);

            Albuns *sabeAlbum;
            if(strcmp(nomeArtista, verifica->artista1) == 0){
               sabeAlbum = BuscarAlbum(&(verifica->albuns1), nomeAlbum);

               if(sabeAlbum == NULL){
               printf("Album nao encontrado\n");
               }
               else{
                  printf("INFORMACOES DO ALBUM:\n");
                  printf("Nome: %s\n", sabeAlbum->tituloAlbum1);
                  printf("Ano: %d\n", sabeAlbum->ano1);
                  printf("Quantidade de Musicas: %d\n", sabeAlbum-
>quantmusica1);
                  printf("Artista(s): %s\n", verifica->artista1);
                  printf("\n");
               }

            }

            else if(strcmp(nomeArtista, verifica->artista2) == 0){

               sabeAlbum = BuscarAlbum(&(verifica->albuns2), nomeAlbum);
               if(sabeAlbum == NULL){
               printf("Album nao encontrado\n");
               }
               else{
                  printf("INFORMACOES DO ALBUM:\n");
                  printf("Nome: %s\n", sabeAlbum->tituloAlbum2);
                  printf("Ano: %d\n", sabeAlbum->ano2);
                  printf("Quantidade de Musicas: %d\n", sabeAlbum-
>quantmusica2);
                  printf("Artista(s): %s\n", verifica->artista2);
                  printf("\n");
               }

            }
```

```
            }

            break;
        }
        case 8:{
            char nomeArtista[200];
            char nomeAlbum[200];
            char nomeMusica[200];
            printf("Informe o Nome do Artista:\n");
            fflush(stdin);
            scanf("%[^\n]", nomeArtista);

            Artista *verifica;
            Musica *BuscaM;
            verifica = BuscaArtista(&raiz, nomeArtista);

            if(verifica == NULL){
                printf("Artista nao encontrado\n");
            }
            else{
                printf("Informe o Nome do Album:\n");
                fflush(stdin);
                scanf("%[^\n]", nomeAlbum);

                Albuns *sabeAlbum;
                if(strcmp(nomeArtista, verifica->artista1) == 0){
                    sabeAlbum = BuscarAlbum(&(verifica->albuns1), nomeAlbum);

                    if(sabeAlbum == NULL){
                    printf("Album nao encontrado\n");
                    }
                    else{
                        printf("Informe o Nome da Musica:\n");
                        fflush(stdin);
                        scanf("%[^\n]", nomeAlbum);
                       BuscaM =  BuscaMusica(sabeAlbum->musica1, nomeMusica);

                        if(BuscaM == NULL){
                            printf("Musica Nao Encontrada no Album %s do Artista %s\n", sabeAlbum->tituloAlbum1, verifica->artista1);
                        }
                        else{
```

```
                    printf("Informacoes da Musica:\n");
                    ImprimeMusicas(BuscaM);
                    printf("Nome do Artista: %s\n", verifica->artista1);
                    printf("Album: %s\n",  sabeAlbum->tituloAlbum1);
                }
            }

        }

        else if(strcmp(nomeArtista, verifica->artista2) == 0){
            sabeAlbum = BuscarAlbum(&(verifica->albuns2), nomeAlbum);

            if(sabeAlbum == NULL){
            printf("Album nao encontrado\n");
            }
            else{
                printf("Informe o Nome da Musica:\n");
                fflush(stdin);
                scanf("%[^\n]", nomeAlbum);
                BuscaM = BuscaMusica(sabeAlbum->musica2, nomeMusica);

                if(BuscaM == NULL){
                    printf("Musica Nao Encontrada no Album %s do Artista %s\n", sabeAlbum->tituloAlbum2, verifica->artista2);
                }
                else{
                    printf("Informacoes da Musica:\n");
                    ImprimeMusicas(BuscaM);
                    printf("Nome do Artista: %s\n", verifica->artista2);
                    printf("Album: %s\n",  sabeAlbum->tituloAlbum2);
                }
            }

        }




    }

    break;
}
```

```
case 9:{
    char nomeArtista[200];
    char nomeAlbum[200];
    char nomeMusica[200];
    printf("Informe o Nome do Artista:\n");
    fflush(stdin);
    scanf("%[^\n]", nomeArtista);

    Artista *verifica;

    verifica = BuscaArtista(&raiz, nomeArtista);

    if(verifica == NULL){
       printf("Artista nao encontrado\n");
    }
    else{
       printf("Informe o Nome do Album:\n");
       fflush(stdin);
       scanf("%[^\n]", nomeAlbum);

       Albuns *sabeAlbum;
       if(strcmp(nomeArtista, verifica->artista1) == 0){
          sabeAlbum = BuscarAlbum(&(verifica->albuns1), nomeAlbum);

          if(sabeAlbum == NULL){
          printf("Album nao encontrado\n");
          }
          else{
             printf("Informe o Nome da Musica:\n");
             fflush(stdin);
             scanf("%[^\n]", nomeAlbum);
             remover_musica(&(sabeAlbum->musica1), nomeMusica);

          }

       }

       else if(strcmp(nomeArtista, verifica->artista2) == 0){
          sabeAlbum = BuscarAlbum(&(verifica->albuns2), nomeAlbum);

          if(sabeAlbum == NULL){
          printf("Album nao encontrado\n");
          }
```

```
                else{
                    printf("Informe o Nome da Musica:\n");
                    fflush(stdin);
                    scanf("%[^\n]", nomeAlbum);
                    remover_musica(&(sabeAlbum->musica2), nomeMusica);

                }

            }




        }

        break;

    }


  }

}

liberarArtista(&raiz);
printf("Memoria Liberada\n");

return 0;
}
```