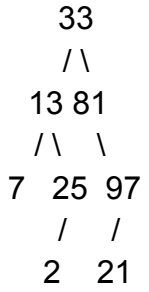
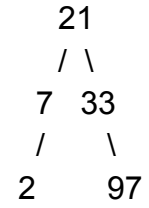
1. Exercício:

a)



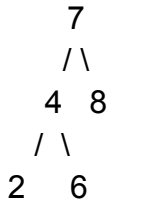
b)



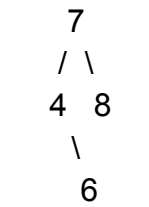
c) Caminhamento central (in-ordem): **2, 7, 21, 33, 97**
d) Caminhamento pós-ordem (post-ordem): **2, 7, 97, 33, 21**
e) Caminhamento pré-ordem (pre-ordem): **21, 7, 2, 33, 97**

2. Exercício:

a)



b)



c) Caminhamento central (in-ordem): **4, 6, 7, 8**

d) Caminhamento pós-ordem (post-ordem): **6, 4, 8, 7**
e) Caminhamento pré-ordem (pre-ordem): **7, 4, 6, 8**

3. Exercício:

```
using System;

class Paciente
{
    public int Cpf { get; set; }
    public string Nome { get; set; }
    public string Email { get; set; }
    public string Telefone { get; set; }
}

class NoAVL
{
    public Paciente Dados { get; set; }
    public int Altura { get; set; }
    public NoAVL Esquerdo { get; set; }
    public NoAVL Direito { get; set; }

    public NoAVL(Paciente dados)
    {
        Dados = dados;
        Altura = 1;
    }
}

class ArvoreAVL
{
    private NoAVL Raiz;

    private int Altura(NoAVL no) => no?.Altura ?? 0;
```

```
    private int FatorBalanceamento(NoAVL no) => no == null ? 0 :
Altura(no.Esquerdo) - Altura(no.Direito);

    private NoAVL RotacaoDireita(NoAVL y)
    {
        var x = y.Esquerdo;
        var T = x.Direito;

        x.Direito = y;
        y.Esquerdo = T;

        y.Altura = Math.Max(Altura(y.Esquerdo), Altura(y.Direito)) + 1;
        x.Altura = Math.Max(Altura(x.Esquerdo), Altura(x.Direito)) + 1;

        return x;
    }

    private NoAVL RotacaoEsquerda(NoAVL x)
    {
        var y = x.Direito;
        var T = y.Esquerdo;

        y.Esquerdo = x;
        x.Direito = T;

        x.Altura = Math.Max(Altura(x.Esquerdo), Altura(x.Direito)) + 1;
        y.Altura = Math.Max(Altura(y.Esquerdo), Altura(y.Direito)) + 1;

        return y;
    }

    private NoAVL Balancear(NoAVL no)
    {
        var fator = FatorBalanceamento(no);

        if (fator > 1)
        {
```

```
            if (FatorBalanceamento(no.Esquerdo) < 0)
                no.Esquerdo = RotacaoEsquerda(no.Esquerdo);
            return RotacaoDireita(no);
        }

        if (fator < -1)
        {
            if (FatorBalanceamento(no.Direito) > 0)
                no.Direito = RotacaoDireita(no.Direito);
            return RotacaoEsquerda(no);
        }

        return no;
    }

    private NoAVL Inserir(NoAVL no, Paciente paciente)
    {
        if (no == null)
            return new NoAVL(paciente);

        if (paciente.Cpf < no.Dados.Cpf)
            no.Esquerdo = Inserir(no.Esquerdo, paciente);
        else if (paciente.Cpf > no.Dados.Cpf)
            no.Direito = Inserir(no.Direito, paciente);
        else
            throw new Exception("Paciente já cadastrado!");

        no.Altura = 1 + Math.Max(Altura(no.Esquerdo), Altura(no.Direito));

        return Balancear(no);
    }

    public void Inserir(Paciente paciente) => Raiz = Inserir(Raiz,
paciente);

    private NoAVL Remover(NoAVL no, int cpf)
    {
```

```
    if (no == null)
        return null;

    if (cpf < no.Dados.Cpf)
        no.Esquerdo = Remover(no.Esquerdo, cpf);
    else if (cpf > no.Dados.Cpf)
        no.Direito = Remover(no.Direito, cpf);
    else
    {
        if (no.Esquerdo == null || no.Direito == null)
            return no.Esquerdo ?? no.Direito;

        var sucessor = Minimo(no.Direito);
        no.Dados = sucessor.Dados;
        no.Direito = Remover(no.Direito, sucessor.Dados.Cpf);
    }

    no.Altura = 1 + Math.Max(Altura(no.Esquerdo), Altura(no.Direito));

    return Balancear(no);
}

public void Remover(int cpf) => Raiz = Remover(Raiz, cpf);

private NoAVL Minimo(NoAVL no)
{
    var atual = no;
    while (atual.Esquerdo != null)
        atual = atual.Esquerdo;
    return atual;
}

private Paciente Buscar(NoAVL no, int cpf)
{
    if (no == null)
        return null;
```

```
        if (cpf < no.Dados.Cpf)
            return Buscar(no.Esquerdo, cpf);
        if (cpf > no.Dados.Cpf)
            return Buscar(no.Direito, cpf);

        return no.Dados;
    }

    public Paciente Buscar(int cpf) => Buscar(Raiz, cpf);

    private void CaminhamentoCentral(NoAVL no, Action<Paciente>
acao)
    {
        if (no == null) return;

        CaminhamentoCentral(no.Esquerdo, acao);
        acao(no.Dados);
        CaminhamentoCentral(no.Direito, acao);
    }

    public void CaminhamentoCentral(Action<Paciente> acao) =>
CaminhamentoCentral(Raiz, acao);

    private void CaminhamentoPosOrdem(NoAVL no, Action<Paciente>
acao)
    {
        if (no == null) return;

        CaminhamentoPosOrdem(no.Esquerdo, acao);
        CaminhamentoPosOrdem(no.Direito, acao);
        acao(no.Dados);
    }

    public void CaminhamentoPosOrdem(Action<Paciente> acao) =>
CaminhamentoPosOrdem(Raiz, acao);
```

```
    private void CaminhamentoPreOrdem(NoAVL no, Action<Paciente>
acao)
    {
        if (no == null) return;

        acao(no.Dados);
        CaminhamentoPreOrdem(no.Esquerdo, acao);
        CaminhamentoPreOrdem(no.Direito, acao);
    }

    public void CaminhamentoPreOrdem(Action<Paciente> acao) =>
CaminhamentoPreOrdem(Raiz, acao);
}

class Program
{
    static void Main()
    {
        var arvore = new ArvoreAVL();

        while (true)
        {
            Console.WriteLine("\n1- Cadastrar um paciente");
            Console.WriteLine("2- Remover um paciente");
            Console.WriteLine("3- Pesquisar um paciente pelo CPF");
            Console.WriteLine("4- Mostrar os nomes de todos os pacientes
(caminhamento central)");
            Console.WriteLine("5- Mostrar os CPFs de todos os pacientes
(caminhamento pós-ordem)");
            Console.WriteLine("6- Mostrar os e-mails de todos os pacientes
(caminhamento pré-ordem)");
            Console.WriteLine("7- Sair");
            Console.Write("Escolha uma opção: ");

            if (!int.TryParse(Console.ReadLine(), out var opcao) || opcao ==
7)
                break;
```

```
        switch (opcao)
        {
            case 1:
                Console.Write("Digite o CPF: ");
                var cpf = int.Parse(Console.ReadLine());
                Console.Write("Digite o Nome: ");
                var nome = Console.ReadLine();
                Console.Write("Digite o Email: ");
                var email = Console.ReadLine();
                Console.Write("Digite o Telefone: ");
                var telefone = Console.ReadLine();
                arvore.Inserir(new Paciente { Cpf = cpf, Nome = nome,
Email = email, Telefone = telefone });
                Console.WriteLine("Paciente cadastrado com sucesso!");
                break;

            case 2:
                Console.Write("Digite o CPF do paciente a ser removido: ");
                cpf = int.Parse(Console.ReadLine());
                arvore.Remover(cpf);
                Console.WriteLine("Paciente removido com sucesso!");
                break;

            case 3:
                Console.Write("Digite o CPF do paciente a ser pesquisado:
");
                cpf = int.Parse(Console.ReadLine());
                var paciente = arvore.Buscar(cpf);
                if (paciente == null)
                    Console.WriteLine("Paciente não cadastrado.");
                else
                    Console.WriteLine($"CPF: {paciente.Cpf}, Nome:
{paciente.Nome}, Email: {paciente.Email}, Telefone:
{paciente.Telefone}");
                break;
```

```
                case 4:
                    Console.WriteLine("Nomes dos pacientes:");
                    arvore.CaminhamentoCentral(p =>
Console.WriteLine(p.Nome));
                    break;

                case 5:
                    Console.WriteLine("CPFs dos pacientes:");
                    arvore.CaminhamentoPosOrdem(p =>
Console.WriteLine(p.Cpf));
                    break;

                case 6:
                    Console.WriteLine("Emails dos pacientes:");
                    arvore.CaminhamentoPreOrdem(p =>
Console.WriteLine(p.Email));
                    break;

                default:
                    Console.WriteLine("Opção inválida!");
                    break;
            }
        }
    }
}
```

4. Exercício:

Representação gráfica da tabela hash:
Índice 0: [ ]
Índice 1: [28 → 19 → 10]
Índice 2: [20]
Índice 3: [12]
Índice 4: [ ]
Índice 5: [5]
Índice 6: [15 → 33]

Índice 7: [ ]
Índice 8: [17]

5. Exercício:

```
using System;
    public class Estudante
    {
        public int Matricula { get; set; }
        public string Nome { get; set; }
        public string Curso { get; set; }

        public Estudante(int matricula, string nome, string curso)
        {
            Matricula = matricula;
            Nome = nome;
            Curso = curso;
        }

        public override string ToString()
        {
            return $"Matrícula: {Matricula}, Nome: {Nome}, Curso: {Curso}";
        }
    }
    public class NoLista
    {
        public Estudante Estudante { get; set; }
        public NoLista Proximo { get; set; }

        public NoLista(Estudante estudante)
        {
            Estudante = estudante;
            Proximo = null;
        }
    }
    public class Lista
    {
```

```
private NoLista primeiro;

public Lista()
{
    primeiro = null;
}

public void Inserir(Estudante estudante)
{
    NoLista novo = new NoLista(estudante);
    novo.Proximo = primeiro;
    primeiro = novo;
}

public Estudante Pesquisar(int matricula)
{
    NoLista atual = primeiro;
    while (atual != null)
    {
        if (atual.Estudante.Matricula == matricula)
            return atual.Estudante;
        atual = atual.Proximo;
    }
    return null;
}

public bool Remover(int matricula)
{
    NoLista atual = primeiro, anterior = null;

    while (atual != null)
    {
        if (atual.Estudante.Matricula == matricula)
        {
            if (anterior == null)
                primeiro = atual.Proximo;
            else
```

```
                anterior.Proximo = atual.Proximo;

                return true;
            }
            anterior = atual;
            atual = atual.Proximo;
        }

        return false;
    }

    public void Listar()
    {
        NoLista atual = primeiro;
        while (atual != null)
        {
            Console.WriteLine(atual.Estudante);
            atual = atual.Proximo;
        }
    }
}
public class TabelaHash
{
    private Lista[] tabela;
    private int tamanho;

    public TabelaHash(int tamanho)
    {
        this.tamanho = tamanho;
        tabela = new Lista[tamanho];
        for (int i = 0; i < tamanho; i++)
        {
            tabela[i] = new Lista();
        }
    }

    private int Hash(int chave)
```

```
    {
        return chave % tamanho;
    }

    public void Inserir(Estudante estudante)
    {
        int posicao = Hash(estudante.Matricula);
        tabela[posicao].Inserir(estudante);
    }

    public Estudante Pesquisar(int matricula)
    {
        int posicao = Hash(matricula);
        return tabela[posicao].Pesquisar(matricula);
    }

    public bool Remover(int matricula)
    {
        int posicao = Hash(matricula);
        return tabela[posicao].Remover(matricula);
    }
}

class Program
{
    static void Main(string[] args)
    {
        TabelaHash tabela = new TabelaHash(101);

        int opcao;
        do
        {
            Console.WriteLine("1- Inserir um estudante");
            Console.WriteLine("2- Remover um estudante");
            Console.WriteLine("3- Pesquisar um estudante");
            Console.WriteLine("4- Sair");
            Console.Write("Escolha uma opção: ");
```

```
            opcao = int.Parse(Console.ReadLine());

            switch (opcao)
            {
                case 1:
                    Console.Write("Informe a matrícula: ");
                    int matricula = int.Parse(Console.ReadLine());
                    Console.Write("Informe o nome: ");
                    string nome = Console.ReadLine();
                    Console.Write("Informe o curso: ");
                    string curso = Console.ReadLine();
                    tabela.Inserir(new Estudante(matricula, nome, curso));
                    Console.WriteLine("Estudante inserido com sucesso!");
                    break;

                case 2:
                    Console.Write("Informe a matrícula do estudante a
remover: ");
                    matricula = int.Parse(Console.ReadLine());
                    if (tabela.Remover(matricula))
                        Console.WriteLine("Estudante removido com
sucesso!");
                    else
                        Console.WriteLine("Estudante não encontrado.");
                    break;

                case 3:
                    Console.Write("Informe a matrícula do estudante a
pesquisar: ");
                    matricula = int.Parse(Console.ReadLine());
                    Estudante estudante = tabela.Pesquisar(matricula);
                    if (estudante != null)
                        Console.WriteLine(estudante);
                    else
                        Console.WriteLine("Estudante não cadastrado.");
                    break;
```

```
                case 4:
                    Console.WriteLine("Saindo...");
                    break;

                default:
                    Console.WriteLine("Opção inválida.");
                    break;
            }

        } while (opcao != 4);
    }
}
```

6. Exercício:

Representação Visual:
Índice: **0**
1 2 3 4 5 6 7 8 9
10
Chave: **22 88 -- -- 4 15 28 17 59 31 10**

7. Exercício:

```
using System;

class TabelaHash
{
    private int[] tabela;
    private int tamanho;

    public TabelaHash(int tamanho)
    {
        this.tamanho = tamanho;
        tabela = new int[tamanho];
        for (int i = 0; i < tamanho; i++)
        {
```

```
        tabela[i] = -1;
    }
}

private int Hash(int chave, int i)
{
    return (chave + i) % tamanho;
}

public bool Inserir(int chave)
{
    for (int i = 0; i < tamanho; i++)
    {
        int posicao = Hash(chave, i);
        if (tabela[posicao] == -1 || tabela[posicao] == -2)
        {
            tabela[posicao] = chave;
            return true;
        }
    }
    Console.WriteLine("Tabela cheia! Não foi possível inserir.");
    return false;
}

public bool Remover(int chave)
{
    for (int i = 0; i < tamanho; i++)
    {
        int posicao = Hash(chave, i);
        if (tabela[posicao] == chave)
        {
            tabela[posicao] = -2;
            return true;
        }
        if (tabela[posicao] == -1)
            break;
    }
```

```
        return false;
    }

    public bool Pesquisar(int chave)
    {
        for (int i = 0; i < tamanho; i++)
        {
            int posicao = Hash(chave, i);
            if (tabela[posicao] == chave)
                return true;
            if (tabela[posicao] == -1)
                break;
        }
        return false;
    }

    public void ExibirTabela()
    {
        Console.WriteLine("Tabela Hash:");
        for (int i = 0; i < tamanho; i++)
        {
            Console.WriteLine($"Posição {i}: {(tabela[i] == -1 ? "Vazia" :
tabela[i] == -2 ? "Removido" : tabela[i].ToString())}");
        }
    }
}

class Program
{
    static void Main(string[] args)
    {
        TabelaHash tabela = new TabelaHash(13);
        int opcao;

        do
        {
            Console.WriteLine("\nMenu de opções:");
```

```
Console.WriteLine("1- Inserir um número");
Console.WriteLine("2- Remover um número");
Console.WriteLine("3- Pesquisar um número");
Console.WriteLine("4- Sair");
Console.Write("Escolha uma opção: ");
opcao = int.Parse(Console.ReadLine());

switch (opcao)
{
    case 1:
        Console.Write("Informe o número a ser inserido: ");
        int numInserir = int.Parse(Console.ReadLine());
        if (tabela.Inserir(numInserir))
            Console.WriteLine("Número inserido com sucesso.");
        else
            Console.WriteLine("Falha ao inserir o número.");
        break;

    case 2:
        Console.Write("Informe o número a ser removido: ");
        int numRemover = int.Parse(Console.ReadLine());
        if (tabela.Remover(numRemover))
            Console.WriteLine("Número removido com sucesso.");
        else
            Console.WriteLine("Número não encontrado.");
        break;

    case 3:
        Console.Write("Informe o número a ser pesquisado: ");
        int numPesquisar = int.Parse(Console.ReadLine());
        if (tabela.Pesquisar(numPesquisar))
            Console.WriteLine("Número encontrado na tabela.");
        else
            Console.WriteLine("Número não encontrado.");
        break;

    case 4:
```

```
                Console.WriteLine("Saindo...");
                break;

            default:
                Console.WriteLine("Opção inválida!");
                break;
        }

        tabela.ExibirTabela();

    } while (opcao != 4);
  }
}
```