Estruturas Ligadas

? EXERCÍCIO 1: INSERÇÃO À CAUDA DE UMA LISTA LIGADA

```
using System;
namespace EL1
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome;}}

  public NoDaLista PProx
   {get
    {return Proximo; }
    set
     {Proximo=value;}}}

 public class Lista
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Lista()
    {PrimeiroNo = UltimoNo = null;}
   public void InserirACauda(string PessoaAInserir)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
     UltimoNo = UltimoNo.PProx=new NoDaLista(PessoaAInserir);}
   public bool ListaVazia()
    {return PrimeiroNo == null; }}

 public class InsercoesACauda
  {static void Main(string[] args)
   {Lista L = new Lista();
    Console.Write("Digite um nome ou ZZZ para terminar ");
    string NovoNome=Console.ReadLine();
    while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
     {L.InserirACauda(NovoNome);
```

```
        Console.Write("Digite um nome ou ZZZ para terminar ");
        NovoNome = Console.ReadLine();}}}}
```

? EXERCÍCIO 2: IMPRESSÃO DOS ELEMENTOS DE UMA LISTA LIGADA

```
using System;
namespace EL2
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
     {return Nome;}}
  public NoDaLista PProx
   {get
     {return Proximo; }
    set
     {Proximo = value; }}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public void InserirACauda(string PessoaAInserir)
   {if (ListaVazia())
    PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
    else
     UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void ImprimirLista()
   {if (ListaVazia())
    {Console.WriteLine("Lista Vazia"); }
     else
      {NoDaLista Corrente = PrimeiroNo;
   while (Corrente != null)
    {Console.WriteLine(Corrente.PNome);
     Corrente = Corrente.PProx;}}}}
```

```
 public class ImpressaoLista
 {static void Main(string[] args)
  {Lista L = new Lista();
   Console.Write("Digite um nome ou ZZZ para terminar ");
   string NovoNome = Console.ReadLine();
   while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
    {L.InserirACauda(NovoNome);
     Console.Write("Digite um nome ou ZZZ para terminar ");
     NovoNome = Console.ReadLine();}
   L.ImprimirLista();}}}
```

? EXERCÍCIO 3: CONTAGEM DE ELEMENTOS DE UMA LISTA LIGADA

```
using System;
namespace EL3
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
     {return Nome;}}
  public NoDaLista PProx
   {get
     {return Proximo; }
    set
     {Proximo = value; }}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public void InserirACauda(string PessoaAInserir)
   {if (ListaVazia())
    PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
    else
     UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
  public bool ListaVazia()
   {return PrimeiroNo == null; }
public int ContagemDeNos()
 {int Contador = 0;
```

```
  if (ListaVazia()==false)
  {NoDaLista Corrente = PrimeiroNo;
   while (Corrente != null)
     {Contador++;
      Corrente = Corrente.PProx;}}
  return Contador;}}

 public class ContagemDeNos
 {static void Main(string[] args)
  {Lista L = new Lista();
   Console.Write("Digite um nome ou ZZZ para terminar ");
   string NovoNome = Console.ReadLine();
   while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
    {L.InserirACauda(NovoNome);
     Console.Write("Digite um nome ou ZZZ para terminar ");
     NovoNome = Console.ReadLine();}
   Console.WriteLine("A lista tem {0} elementos ",
   L.ContagemDeNos());}}}
```

? EXERCÍCIO 4: REMOÇÃO À CABEÇA DE UMA LISTA LIGADA

```
using System;
namespace EL4
{public class NoDaLista

 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
     {return Nome;}}
  public NoDaLista PProx
   {get
     {return Proximo; }
    set
     {Proximo = value; }}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
```

```
    {PrimeiroNo = UltimoNo = null; }
   public void InserirACauda(string PessoaAInserir)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
      UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
   public bool ListaVazia()
    {return PrimeiroNo == null; }
public string RemocaoACabeca()
 {string NomeEliminado = "";
   if (ListaVazia())
    Console.WriteLine("A lista está vazia");
    else
     {NomeEliminado = PrimeiroNo.PNome;
      if (PrimeiroNo == UltimoNo)
       PrimeiroNo = UltimoNo = null;
       else
        PrimeiroNo = PrimeiroNo.PProx;}
  return NomeEliminado;}
public void ImprimirLista()
 {if (ListaVazia())
   {Console.WriteLine("Lista Vazia"); }
    else
     {NoDaLista Corrente = PrimeiroNo;
      while (Corrente != null)
       {Console.WriteLine(Corrente.PNome);
        Corrente = Corrente.PProx;}}}}
```

```
 public class RemocaoCabeca
 {static void Main(string[] args)
  {Lista L = new Lista();
   Console.Write("Digite um nome ou ZZZ para terminar ");
   string NovoNome = Console.ReadLine();
   while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
    {L.InserirACauda(NovoNome);
     Console.Write("Digite um nome ou ZZZ para terminar ");
     NovoNome = Console.ReadLine();}
     Console.WriteLine ("Antes da remoção:");
     L.ImprimirLista();
     Console.WriteLine("Eliminamos o(a) {0}",
                        L.RemocaoACabeca());
     Console.WriteLine ("Depois da remoção:");
     L.ImprimirLista();}}}
```

? EXERCÍCIO 5: INSERÇÃO À CABEÇA DE UMA LISTA LIGADA

```
using System;
namespace EL5
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome; }}
   public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value;}}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void InsercaoACabeca(string PessoaAInserir)
   {if (ListaVazia())
    PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
    else
    PrimeiroNo = new NoDaLista(PessoaAInserir, PrimeiroNo);}
  public void ImprimirLista()
   {if (ListaVazia()==true)
    {Console.WriteLine("Lista Vazia"); }
    else
    {NoDaLista Corrente = PrimeiroNo;
    while (Corrente != null)
     {Console.WriteLine(Corrente.PNome);
      Corrente = Corrente.PProx;}}}}

public class InsercoesACabeca
 {static void Main(string[] args)
  {Lista L = new Lista();
   L.InsercaoACabeca("Maria");
```

```
    L.InsercaoACabeca("Joana");
    L.InsercaoACabeca("Rita");
    L.InsercaoACabeca("Eva");
    L.ImprimirLista();}}}
```

? EXERCÍCIO 6: REMOÇÃO À CAUDA DE UMA LISTA LIGADA

```
using System;
namespace EL6
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
  {Nome = Pessoa;
   Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
 public string PNome
  {get
   {return Nome;}}
 public NoDaLista PProx
  {get
   {return Proximo;}
  set
   {Proximo = value;}}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void InserirACabeca(string PessoaAInserir)
   {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
     PrimeiroNo = new NoDaLista(PessoaAInserir, PrimeiroNo);}
   public void RemocaoACauda()
    {if (ListaVazia()==true)
      Console.WriteLine("A lista está vazia.");
      else
      if (PrimeiroNo == UltimoNo)
       PrimeiroNo = UltimoNo = null;
       else
```

```
          {NoDaLista Corrente = PrimeiroNo;
          while (Corrente.PProx != UltimoNo)
           Corrente = Corrente.PProx;
          UltimoNo = Corrente;
          Corrente.PProx = null;}}
    public void ImprimirLista()
    {if (ListaVazia())
    {Console.WriteLine("Lista Vazia"); }
    else
    {NoDaLista Corrente = PrimeiroNo;
    while (Corrente != null)
    {Console.WriteLine(Corrente.PNome);
    Corrente = Corrente.PProx;}}}}

public class RemocoesACauda
 {static void Main(string[] args)
    {string[] Nomes = { "Maria", "Joana", "Rita", "Eva" };
     Lista L = new Lista();
     for (int I=0; I<Nomes.Length; I++)
       L.InserirACabeca(Nomes[I]);
    Console.WriteLine("Antes da remoção:");
    L.ImprimirLista();
    L.RemocaoACauda();
    Console.WriteLine("Depois da remoção:");
    L.ImprimirLista();}}}
```

? EXERCÍCIO 7: REMOÇÃO DE UM ELEMENTO DE UMA LISTA LIGADA

```
using System;
namespace EL7
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
   Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
   Proximo = Prox;}
  public string PNome
   {get
    {return Nome; }}
  public NoDaLista PProx
   {get
    {return Proximo; }
    set
```

```
    {Proximo = value; }}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void InserirACabeca(string PessoaAInserir)
  {if (ListaVazia())
   PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
   else
    PrimeiroNo = new NoDaLista(PessoaAInserir, PrimeiroNo);}
  public void RemocaoDeNome(string PessoaARemover)
  {if (ListaVazia() == true)
   Console.WriteLine("A lista está vazia.");
   else
    {bool Enc = false;
    NoDaLista Corrente = PrimeiroNo;
    if (Corrente.PNome == PessoaARemover)
     {PrimeiroNo=Corrente.PProx ;
      Corrente=null;}
      else
      {NoDaLista Anterior = Corrente;
      Corrente = Corrente.PProx;
      while ((Corrente != null) && (Enc == false))
       {if (Corrente.PNome == PessoaARemover)
        Enc = true;
        else
        {Anterior = Corrente;
         Corrente = Corrente.PProx;}}
      if (Enc == true)
       {Anterior.PProx = Corrente.PProx;
        Corrente = null;}}}}
  public void ImprimirLista()
   {if (ListaVazia()==true)
    Console.WriteLine("Lista Vazia");
    else
     {NoDaLista Corrente = PrimeiroNo;
      while (Corrente != null)
       {Console.WriteLine(Corrente.PNome);
        Corrente = Corrente.PProx;}}}}

public class RemocoesDeNomes
 {static void Main(string[] args)
  {Lista L = new Lista();
   L.InserirACabeca("Maria");
   L.InserirACabeca("Joana");
```

```
    L.InserirACabeca("Rita");
    L.InserirACabeca("Eva");
    Console.WriteLine("Antes da remoção:");
    L.ImprimirLista();
    L.RemocaoDeNome("Joana");
    Console.WriteLine("Depois da remoção:");
    L.ImprimirLista();}}}
```

? EXERCÍCIO 8: INSERÇÃO DE UM ELEMENTO NUMA LISTA LIGADA

```
using System;
namespace EL8
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
   {return Nome; }}
  public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value; }}}

  public class Lista
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Lista()
    {PrimeiroNo = UltimoNo = null; }
   public bool ListaVazia()
    {return PrimeiroNo == null; }
   public void InserirACabeca(string PessoaAInserir)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
     PrimeiroNo = new NoDaLista(PessoaAInserir, PrimeiroNo);}
   public int Tamanho()
   {int Contador = 0;
    if (ListaVazia()==false)
     {NoDaLista Corrente = PrimeiroNo;
```

```
    while (Corrente != null)
     {Contador++;
      Corrente = Corrente.PProx;}}
  return Contador;}
 public void InsercaoNaPosicaoK(string PessoaAInserir, int K)
 {K = Math.Min(K, this.Tamanho() + 1);
  NoDaLista Novo = new NoDaLista(PessoaAInserir);
  if (ListaVazia() == true)
   PrimeiroNo = UltimoNo = Novo;
   else if (K == 0)
    {Novo.PProx = PrimeiroNo;
     PrimeiroNo = Novo;}
     else
      {int N = 1;
        NoDaLista Corrente = PrimeiroNo;
        while (N < K-1)
         {Corrente = Corrente.PProx;
          N++;}
        Novo.PProx = Corrente.PProx;
        Corrente.PProx = Novo;}}
 public void ImprimirLista()
  {if (ListaVazia()==true)
   {Console.WriteLine("Lista Vazia"); }
    else
     {NoDaLista Corrente = PrimeiroNo;
      while (Corrente != null)
      {Console.WriteLine(Corrente.PNome);
       Corrente = Corrente.PProx;}}}}

public class InsercaoNaPosicaoK
 {static void Main(string[] args)
  {Lista L = new Lista();
   L.InserirACabeca("Maria");
   L.InserirACabeca("Joana");
   L.InserirACabeca("Rita");
   L.InserirACabeca("Eva");
   Console.WriteLine("Antes da inserção:");
   L.ImprimirLista();
   int K = 3;
   K = Math.Min(K, L.Tamanho()+1);
   L.InsercaoNaPosicaoK("Carla", K);
   Console.WriteLine("Depois da inserção:");
   L.ImprimirLista();}}}
```

? EXERCÍCIO 9: INSERÇÃO POR ORDEM ALFABÉTICA NUMA LISTA LIGADA

```
using System;
namespace EL9
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome;}}
  public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value;}}}

 public class Lista
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Lista()
    {PrimeiroNo = UltimoNo = null; }
   public bool ListaVazia()
    {return PrimeiroNo == null; }
   public void InserirACauda(string PessoaAInserir)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
     UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
   public void InserirAlfabetica(string PessoaAInserir)
    {NoDaLista Novo = new NoDaLista(PessoaAInserir);
     if (ListaVazia() == true)
      PrimeiroNo = UltimoNo = Novo;
      else
      {if (PrimeiroNo.PNome.CompareTo(PessoaAInserir) > 0)
       {Novo.PProx = PrimeiroNo;
        PrimeiroNo = Novo;}
        else
        {NoDaLista Anterior = PrimeiroNo;
         NoDaLista Corrente = Anterior.PProx;
         while (Corrente != null &&
         Corrente.PNome.CompareTo(PessoaAInserir) < 0)
```

```
            {Anterior = Corrente;
           Corrente = Corrente.PProx;}
          Anterior.PProx = Novo;
          Novo.PProx = Corrente;}}}

   public void ImprimirLista()
    {if (ListaVazia()==true)
     Console.WriteLine("Lista Vazia");
     else
      {NoDaLista Corrente = PrimeiroNo;
      while (Corrente != null)
       {Console.WriteLine(Corrente.PNome);
        Corrente = Corrente.PProx;}}}}

public class InsercaoAlfabetica
 {static void Main(string[] args)
  {Lista L = new Lista();
   L.InserirACauda("Ana");
   L.InserirACauda("Beatriz");
   L.InserirACauda("Dalila");
   L.InserirACauda("Maria");
   Console.WriteLine("Antes da inserção:");
   L.ImprimirLista();
   L.InserirAlfabetica("Carla");
   Console.WriteLine("Depois da inserção:");
   L.ImprimirLista();}}}
```

? EXERCÍCIO 10: FORMAÇÃO DE LISTA LIGADA COM INSERÇÕES À CAUDA

```
using System;
namespace EL10
{public class NoDaLista
 {private string Nome;
  private int Idade;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa, int X)
   {Nome = Pessoa;
    Idade = X;
    Proximo = null;}
  public NoDaLista(string Pessoa, int X, NoDaLista Prox)
   {Nome = Pessoa;
    Idade = X;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome;}}
```

```
  public int PIdade
   {get
    {return Idade; }}
  public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value; }}}

 public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null;}
  public void InserirACauda(string PessoaAInserir, int X)
   {if (ListaVazia())
    PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir, X);
    else
     UltimoNo=UltimoNo.PProx = new NoDaLista(PessoaAInserir, X);}
  public bool ListaVazia()
   {return PrimeiroNo == null;}}

 public class NosComNomesEIdades
 {static void Main(string[] args)
  {int X;
   Lista L = new Lista();
   Console.Write("Digite um nome ou ZZZ para terminar ");
   string NovoNome = Console.ReadLine();
   while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
    {Console.Write("Digite a idade de "+NovoNome+" ");
     X=Convert.ToInt16(Console.ReadLine());
     L.InserirACauda(NovoNome, X);
     Console.Write("Digite um nome ou ZZZ para terminar ");
     NovoNome = Console.ReadLine();}}}}
```

? EXERCÍCIO 11: PERCURSO DE LISTA LIGADA

```
using System;
namespace EL11
{public class NoDaLista
 {private string Nome;
  private int Idade;
  private NoDaLista Proximo;
 public NoDaLista(string Pessoa, int X)
  {Nome = Pessoa;
   Idade = X;
```

```
    Proximo = null;}
  public NoDaLista(string Pessoa, int X, NoDaLista Prox)
   {Nome = Pessoa;
    Idade = X;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome; }}
   public int PIdade
   {get
    {return Idade; }}
   public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value; }}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public void InserirACauda(string PessoaAInserir, int X)
   {if (ListaVazia())
    PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir, X);
    else
    UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir,
    X);}
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public double MediaIdades()
   {double S = 0;
    int N = 0;
    NoDaLista Corrente = PrimeiroNo;
    while (Corrente != null)
     {N++;
     S += Corrente.PIdade;

     Corrente = Corrente.PProx;}
    return S/N;}}

public class MediasIdades
 {static void Main(string[] args)
 {int X;
  Lista L = new Lista();
  Console.Write("Digite um nome ou ZZZ para terminar ");
  string NovoNome = Console.ReadLine();
  while (NovoNome.ToUpper().CompareTo("ZZZ") != 0)
   {Console.Write("Digite a idade de "+NovoNome+" ");
```

```
    X=Convert.ToInt16(Console.ReadLine());
    L.InserirACauda(NovoNome, X);
    Console.Write("Digite um nome ou ZZZ para terminar ");
    NovoNome = Console.ReadLine();}
  Console.WriteLine("Média das idades {0} anos",
  L.MediaIdades());}}}
```

? EXERCÍCIO 12: ESTRUTURAS EM ELEMENTOS DE UMA LISTA LIGADA

```
using System;
namespace EL12
{public class Estruturas
 {public struct Turma
  {private char desig;
   private int nr;
   private int nrz;
   public Turma(char desig, int nr, int nrz)
    {this.desig = desig;
     this.nr = nr;
     this.nrz = nrz;}
   public char Designacao
    {get
     {return desig; }}
   public int numraparigas
    {get
     {return nr; }}
   public int numrapazes
    {get
     {return nrz; }}
   public override string ToString()
    {return String.Format("A turma {0} tem {1} alunos",
    desig,nr+nrz);}}}

 public class NoDaLista
  {private Estruturas.Turma T;
   private NoDaLista Proximo;
   public NoDaLista(char Tdes, int Ra, int Rz)
    {T = new Estruturas.Turma(Tdes, Ra, Rz);
     Proximo = null;}
   public NoDaLista(char desig, int Ra, int Rz, NoDaLista Prox)
    {T = new Estruturas.Turma(desig, Ra, Rz);
     Proximo = Prox;}
   public NoDaLista PProx
    {get
     {return Proximo; }
    set
```

```
      {Proximo = value; }}

    public override string ToString()
     {return String.Format("A turma {0} tem {1} alunos",
     T.Designacao, T.numraparigas + T.numrapazes);}}

 public class Lista
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Lista()
   {PrimeiroNo = UltimoNo = null; }
   public void InserirACauda(char desig, int Ra, int Rz)
   {if (ListaVazia()== true)
    PrimeiroNo = UltimoNo = new NoDaLista(desig, Ra, Rz);
    else
     UltimoNo = UltimoNo.PProx = new NoDaLista(desig, Ra, Rz);}
   public bool ListaVazia()
   {return PrimeiroNo == null; }
   public void ImprimirLista()
   {if (ListaVazia())

    Console.WriteLine("Lista Vazia");
    else
    {NoDaLista Corrente = PrimeiroNo;
     while (Corrente != null)
     {Console.WriteLine(Corrente.ToString());
      Corrente = Corrente.PProx;}}}}

 public class ElementosComEstruturas
  {static void Main(string[] args)
   {Lista L = new Lista();
    int[,] T={{10,30}, {20,15}, {30,20}};
    for (int i = 65; i <= 65 + 2; i++)
     L.InserirACauda((char)i, T[i-65,0], T[i-65,1]);
    L.ImprimirLista();}}}
```

? EXERCÍCIO 13: MAIS MÉTODOS PARA ESTRUTURA DOS ELEMENTOS DE UMA LISTA LIGADA

```
using System;
namespace EL13
{public class Estruturas
 {public struct Turma
  {private char Desig;
   private int Nr;
   private int Nrz;
```

```
   public Turma(char Desig, int Nr, int Nrz)
    {this.Desig = Desig;
     this.Nr = Nr;
     this.Nrz = Nrz;}
   public char Designacao
    {get
     {return Desig;}}
   public int Numraparigas
    {get
     {return Nr;}}
   public int Numrapazes
    {get
     {return Nrz; }}
   public double Percentagem()
   {int Totalinsc = Nr + Nrz;
    return Math.Round(Nr/ (double) Totalinsc *100,1);}}}

public class NoDaLista
 {private Estruturas.Turma T;
  private NoDaLista Proximo;
  public NoDaLista(char Tdes, int Ra, int Rz)
   {T = new Estruturas.Turma(Tdes, Ra, Rz);
    Proximo = null;}
  public NoDaLista(char desig, int Ra, int Rz, NoDaLista Prox)
   {T = new Estruturas.Turma(desig, Ra, Rz);
    Proximo = Prox;}
  public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value; }}
  public override string ToString()
   {string resultado="";
    resultado = String.Format("A turma {0} tem {1} alunos",
    T.Designacao, T.Numraparigas + T.Numrapazes);
    resultado+="\n";
    resultado+=String.Format("{0}% raparigas e {1}% rapazes",
    T.Percentagem() ,100-T.Percentagem());
    return resultado;}}

public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public void InserirACauda(char desig, int Ra, int Rz)
 {if (ListaVazia())
  PrimeiroNo = UltimoNo = new NoDaLista(desig, Ra, Rz);
  else
```

```
    UltimoNo = UltimoNo.PProx = new NoDaLista(desig, Ra, Rz);}
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void ImprimirLista()
   {if (ListaVazia()==true )
    Console.WriteLine("Lista Vazia");
    else
     {NoDaLista Corrente = PrimeiroNo;
      while (Corrente != null)
       {Console.WriteLine(Corrente.ToString());
        Corrente = Corrente.PProx;}}}}

 public class Percentagens
  {static void Main(string[] args)
   {Lista L = new Lista();
    L.InserirACauda('A', 21, 9);
    L.InserirACauda('B', 50, 10);
    L.ImprimirLista();}}}
```

? EXERCÍCIO 14: DISCIPLINA LIFO PARA LISTA LIGADA

```
using System;
namespace EL14
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome;}}
  public NoDaLista PProx
   {get
    {return Proximo; }
   set
    {Proximo = value; }}}

 public class Certificados
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Certificados()
    {PrimeiroNo = UltimoNo = null; }
```

```
  public void InserirACauda(string PessoaAInserir)
   {if (ListaVazia()==true)
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
      UltimoNo=UltimoNo.PProx=new NoDaLista(PessoaAInserir);}
  public NoDaLista RemocaoACauda()
   {NoDaLista Elim = null;
    if (ListaVazia() != true)
     {if (PrimeiroNo == UltimoNo)
      {Elim = PrimeiroNo;
       PrimeiroNo = UltimoNo = null;}
       else
        {NoDaLista Corrente = PrimeiroNo;
        while (Corrente.PProx != UltimoNo)
         {Corrente = Corrente.PProx; }
          Elim = Corrente.PProx;
          UltimoNo = Corrente;
          Corrente.PProx = null;}}
    return Elim;}
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void ImprimirLista()
   {if (ListaVazia())
    Console.WriteLine("Lista Vazia");
    else
     {NoDaLista Corrente = PrimeiroNo;
     while (Corrente != null)
      {Console.WriteLine("{0}", Corrente.PNome);
       Corrente = Corrente.PProx;}}}}

public class LifoParaCertificados
 {static void Main(string[] args)
  {string[] Nomes = { "Carla", "Teresa", "Rui", "Roberto",
  "Joaquim", "Maria", "Marta" };
   Certificados C = new Certificados();
   for (int I = 0; I <= Nomes.GetLength(0) - 1; I++)
    C.InserirACauda(Nomes[I]);
   Console.WriteLine("Nome do utente");
   NoDaLista Elim = C.RemocaoACauda();
   while (Elim!= null)
    {Console.WriteLine(Elim.PNome);
     Elim = C.RemocaoACauda();}}}}
```

? EXERCÍCIO 15: DISCIPLINA FIFO PARA LISTA LIGADA

```
using System;
namespace EL15
{public class NoDaFila
 {private string Nome;
  private DateTime HoraChegada;
  private NoDaFila Proximo;
   public NoDaFila(string Pessoa, DateTime X)
   {Nome = Pessoa;
    HoraChegada= X;
    Proximo = null;}
  public NoDaFila(string Pessoa, DateTime X, NoDaFila Prox)
  {Nome = Pessoa;
   HoraChegada = X;
   Proximo = Prox;}
  public string PNome
  {get
   {return Nome;}}
  public DateTime PHoraChegada
  {get
   {return HoraChegada;}}
  public NoDaFila PProx
  {get
   {return Proximo; }
  set
   {Proximo = value; }}}

  public class FilaEspera
  {private NoDaFila PrimeiroNo;
   private NoDaFila UltimoNo;
   public FilaEspera()
    {PrimeiroNo = UltimoNo = null; }
   public void InserirACauda(string PessoaAInserir, DateTime X)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaFila(PessoaAInserir, X);
     else
      UltimoNo=UltimoNo.PProx = new NoDaFila(PessoaAInserir, X);}
   public bool ListaVazia()
    {return PrimeiroNo == null; }

  public NoDaFila RemoverACabeca()
   {NoDaFila Eliminado=null;
    if (ListaVazia())
     Console.WriteLine("A lista está vazia");
     else
     {Eliminado = PrimeiroNo;
```

```
      if (PrimeiroNo == UltimoNo)
       PrimeiroNo = UltimoNo = null;
       else
        PrimeiroNo = PrimeiroNo.PProx;}
   return Eliminado;}
  public void ImprimirLista()
   {if (ListaVazia())
    {Console.WriteLine("Lista Vazia");}
     else
      {NoDaFila Corrente = PrimeiroNo;
      while (Corrente != null)
       {Console.WriteLine("{0}\t {1}",Corrente.PNome,
        Corrente.PHoraChegada);
        Corrente = Corrente.PProx;}}}}

public class FifoParaFilasEspera
{static void Main(string[] args)
 {string[] Nomes = {"Carla", "Teresa", "Joana", "Rui", "Pedro",
  "Ana", "Tiago"};
  string[] Horas = { "9:00", "9:03", "9:10", "9:18", "9:20",
  "9:30", "9:32"};
  FilaEspera F = new FilaEspera();
  for (int I = 0; I <= Nomes.GetLength(0) - 1; I++)
   F.InserirACauda(Nomes[I], Convert.ToDateTime(Horas[I]));
  DateTime HoraControlo=Convert.ToDateTime("9:35:00");
  DateTime Corrente=Convert.ToDateTime("9:00:00");
  int Servico=10, SHoras, SMinutos;
  NoDaFila Elim;
  Console.WriteLine("Nome".PadRight(21)+"Entrada".PadRight(24)+
  "Saída");
  while (Corrente <= HoraControlo)
   {Elim=F.RemoverACabeca();
    if (Corrente < Elim.PHoraChegada)
     Corrente = Elim.PHoraChegada;
     SMinutos = Corrente.Minute + Servico;
     SHoras = Corrente.Hour + SMinutos / 60;
     SMinutos = SMinutos % 60;
     Corrente = Convert.ToDateTime(Convert.ToString(SHoras)+":"+
     Convert.ToString(SMinutos));
     Console.WriteLine("{0}\t {1} \t{2}",
     Elim.PNome.PadRight(8), Elim.PHoraChegada, Corrente);}
  Console.WriteLine("Estão ainda na Fila:");
  F.ImprimirLista();}}}
```

? EXERCÍCIO 16: LISTA CIRCULAR

```
using System;
namespace EL16
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
 public NoDaLista(string Pessoa, NoDaLista Prox)
  {Nome = Pessoa;
   Proximo = Prox;}
 public string PNome
  {get
   {return Nome;}}
 public NoDaLista PProx
  {get
   {return Proximo;}
  set
   {Proximo = value;}}}

 public class Lista
 {private NoDaLista PrimeiroNo;
  private NoDaLista UltimoNo;
  public Lista()
   {PrimeiroNo = UltimoNo = null; }
  public bool ListaVazia()
   {return PrimeiroNo == null; }
  public void InserirACauda(string PessoaAInserir)
   {if (ListaVazia())
   PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
   else
   UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
  public int ContagemDeNos()
   {int Contador = 0;
   if (ListaVazia()==false)
    {NoDaLista Corrente = PrimeiroNo;
    while (Corrente != null)
     {Contador++;
      Corrente = Corrente.PProx;}}
   return Contador;}
  public void Circular()
   {UltimoNo.PProx=PrimeiroNo;}
  public void ImprimirListaCircular(int TElementos)
   {if (ListaVazia())
    {Console.WriteLine("Lista Vazia"); }
```

```
      else
      {NoDaLista Corrente = PrimeiroNo;
      for (int I = 1; I <=TElementos; I++)
       {Console.Write(Corrente.PNome+ " ");
        Corrente = Corrente.PProx;}
      Console.WriteLine();}}}

 public class ListasCirculares
 {static void Main(string[] args)
  {Lista L = new Lista();
   string[] Nomes={"Eva", "Rita","Carla","Joana","Maria"};
   for (int I=0; I< Nomes.Length ;I++)
    L.InserirACauda(Nomes[I]);
   int Vezes = 3;
   int TElementos= Vezes* L.ContagemDeNos();
   L.Circular();
   L. ImprimirListaCircular(TElementos);}}}
```

? EXERCÍCIO 17: SELEÇÃO DE ELEMENTOS DE UMA LISTA CIRCULAR

```
using System;
namespace EL17
{public class NoDaLista
 {private string Nome;
  private NoDaLista Proximo;
  public NoDaLista(string Pessoa)
   {Nome = Pessoa;
    Proximo = null;}
  public NoDaLista(string Pessoa, NoDaLista Prox)
   {Nome = Pessoa;
    Proximo = Prox;}
  public string PNome
   {get
    {return Nome; }}
  public NoDaLista PProx
   {get
    {return Proximo;}
   set
    {Proximo = value;}}}

 public class Lista
  {private NoDaLista PrimeiroNo;
   private NoDaLista UltimoNo;
   public Lista()
    {PrimeiroNo = UltimoNo = null; }
   public bool ListaVazia()
```

```
    {return PrimeiroNo == null; }
   public void InserirACauda(string PessoaAInserir)
    {if (ListaVazia())
     PrimeiroNo = UltimoNo = new NoDaLista(PessoaAInserir);
     else
      UltimoNo = UltimoNo.PProx = new NoDaLista(PessoaAInserir);}
   public int ContagemDeNos()
    {int Contador = 0;
     if (ListaVazia()==false)
      {NoDaLista Corrente = PrimeiroNo;
       while (Corrente != null)
        {Contador++;
        Corrente = Corrente.PProx;}}
     return Contador;}
   public void Circular()
    {UltimoNo.PProx=PrimeiroNo;}
   public void PimPamPum(ref int NNos)
    {NoDaLista Corrente = PrimeiroNo;
     Console.WriteLine("Começamos com {0} ", Corrente.PNome);
   while (NNos > 1)
    {Console.WriteLine("Eliminar {0}", Corrente.PProx.PNome);
     Corrente.PProx = Corrente.PProx.PProx;
     Corrente = Corrente.PProx;
     NNos--;}
    PrimeiroNo =Corrente;
    Console.WriteLine("Foi selecionado(a) {0}",
                      PrimeiroNo.PNome);}}

 public class SelecaoPimPamPum
  {static void Main(string[] args)
   {Lista L = new Lista();
    string[] Nomes={"Rui","Ivo","Eva","Joana","Luísa","Maria"};
    for (int I = 0; I < Nomes.Length; I++ )
     L.InserirACauda(Nomes[I]);
    int NNos=L.ContagemDeNos();
    L.Circular();
    L.PimPamPum(ref NNos);}}}
```

? EXERCÍCIO 18: FORMAÇÃO DE ÁRVORE BINÁRIA

```
using System;
namespace EL18
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
```

```
  public NoDaArvore(int D)
   {Dado = D;
   Esquerda = Direita = null;}
  public int PDado
   {get
     {return Dado;}}
  public NoDaArvore PEsquerda
   {get
     {return Esquerda;}}
  public NoDaArvore PDireita
   {get
     {return Direita;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
      Esquerda.InserirRegisto(InteiroParaNo);}
      else if (InteiroParaNo > Dado)
            {if (Direita == null)
             Direita = new NoDaArvore(InteiroParaNo);
             else
              Direita.InserirRegisto(InteiroParaNo);}}}
public class Arvore
 {private NoDaArvore Raiz;
  public Arvore()
   {Raiz = null; }
  public NoDaArvore PRaiz
   {get
     {return Raiz;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}}
public class CriacaoArvore
 {static void Main(string[] args)
  {Arvore T = new Arvore();
   int InteiroParaNo;
   Random R=new Random();
   Console.Write("Números extraídos aleatoriamente: ");
   for (int I = 1; I <10; I++)
    {InteiroParaNo = R.Next(100);
     Console.Write(InteiroParaNo+ " ");
     T.InserirNo(InteiroParaNo);}
    Console.WriteLine("\n A árvore foi criada!");}}}
```

? EXERCÍCIO 19: PERCURSO PRÉ-ORDEM DE ÁRVORE BINÁRIA

```
using System;
namespace EL19
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public int PDado
   {get
     {return Dado;}}
  public NoDaArvore PEsquerda
   {get
     {return Esquerda;}}
  public NoDaArvore PDireita
   {get
     {return Direita;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)

    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
      Esquerda.InserirRegisto(InteiroParaNo);}
      else if (InteiroParaNo > Dado)
       {if (Direita == null)
        Direita = new NoDaArvore(InteiroParaNo);
        else
         Direita.InserirRegisto(InteiroParaNo);}}}

public class Arvore
 {private NoDaArvore Raiz;
  public Arvore()
   {Raiz = null; }
  public NoDaArvore PRaiz
   {get
     {return Raiz;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}
  public void PercursoPreOrdem()
   {PreOrdem(Raiz);}
```

```
  public void PreOrdem(NoDaArvore No)
   {if (No == null)
      return;
      Console.Write(No.PDado + " ");
      PreOrdem(No.PEsquerda);
      PreOrdem(No.PDireita);}}

public class PercursoPreOrdemArvore
 {static void Main(string[] args)
 {Arvore T = new Arvore();
 int InteiroParaNo;
 int [] Nos={13,12,40,31,50,51,48,86,24};
 for (int I = 0; I < Nos.Length; I++)
  {InteiroParaNo = Nos[I];
   T.InserirNo(InteiroParaNo);}
 T.PercursoPreOrdem();
 Console.WriteLine();}}}
```

? EXERCÍCIO 20: PERCURSO EM ORDEM DE ÁRVORE BINÁRIA

```
using System;
namespace EL20
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public int PDado
   {get
     {return Dado;}}
  public NoDaArvore PEsquerda
   {get
     {return Esquerda;}}
  public NoDaArvore PDireita
   {get
     {return Direita;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
      Esquerda.InserirRegisto(InteiroParaNo);}
      else if (InteiroParaNo > Dado)
             {if (Direita == null)
```

```
                Direita = new NoDaArvore(InteiroParaNo);
                else
                 Direita.InserirRegisto(InteiroParaNo);}}}

public class Arvore
 {private NoDaArvore Raiz;
  public Arvore()
   {Raiz = null;}
  public NoDaArvore PRaiz
   {get
     {return Raiz;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}
  public void PercursoEmOrdem()
   {EmOrdem(Raiz);}
  public void EmOrdem(NoDaArvore No)
   {if (No == null)
     return;
     EmOrdem(No.PEsquerda);
     Console.Write(No.PDado + " ");
     EmOrdem(No.PDireita);}}

public class PercursoEmOrdemArvore
 {static void Main(string[] args)
 {Arvore T = new Arvore();
 int InteiroParaNo;
 int [] Nos={13,12,40,31,50,51,48,86,24};
 for (int I = 0; I < Nos.Length; I++)
  {InteiroParaNo = Nos[I];
   T.InserirNo(InteiroParaNo);}
 T.PercursoEmOrdem();
 Console.WriteLine();}}}
```

? EXERCÍCIO 21: PERCURSO PÓS-ORDEM DE ÁRVORE BINÁRIA

```
using System;
namespace EL21
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
```

```
    Esquerda = Direita = null;}
   public int PDado
    {get
      {return Dado;}}
   public NoDaArvore PEsquerda
    {get
      {return Esquerda;}}
   public NoDaArvore PDireita
    {get
      {return Direita;}}
   public void InserirRegisto(int InteiroParaNo)
    {if (InteiroParaNo < Dado)
     {if (Esquerda == null)
      Esquerda = new NoDaArvore(InteiroParaNo);
      else
       Esquerda.InserirRegisto(InteiroParaNo);}
       else if (InteiroParaNo > Dado)
             {if (Direita == null)
              Direita = new NoDaArvore(InteiroParaNo);
              else
               Direita.InserirRegisto(InteiroParaNo);}}}

public class Arvore
 {public NoDaArvore Raiz;
  public Arvore()
   {Raiz = null;}
  public NoDaArvore PRaiz
   {get
     {return Raiz;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}
  public void PercursoPosOrdem()
   {PosOrdem(Raiz);}
  public void PosOrdem(NoDaArvore No)
   {if (No == null)
     return;
     PosOrdem(No.PEsquerda);
     PosOrdem(No.PDireita);
     Console.Write(No.PDado + " ");}}

public class PercursoPosOrdem
 {static void Main(string[] args)
 {Arvore T = new Arvore();
 int InteiroParaNo;
 int [] Nos={13,12,40,31,50,51,48,86,24};
 for (int I = 0; I < Nos.Length; I++)
```

```
  {InteiroParaNo = Nos[I];
   T.InserirNo(InteiroParaNo);}
 T.PercursoPosOrdem();
 Console.WriteLine();}}}
```

? EXERCÍCIO 22: CONTAGEM DE ELEMENTOS DE ÁRVORE BINÁRIA

```
using System;
namespace EL22
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public int PDado
   {get
    {return Dado;}}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
     Esquerda.InserirRegisto(InteiroParaNo);}
     else if (InteiroParaNo > Dado)
           {if (Direita == null)
              Direita = new NoDaArvore(InteiroParaNo);
              else
               Direita.InserirRegisto(InteiroParaNo);}}}

 public class Arvore
  {private NoDaArvore Raiz;
   public Arvore()
    {Raiz = null; }
  public NoDaArvore PRaiz
   {get
     {return Raiz;}}
   public void InserirNo(int InteiroParaNo)
    {if (Raiz == null)
```

```
      Raiz = new NoDaArvore(InteiroParaNo);
      else
       Raiz.InserirRegisto(InteiroParaNo);}
   public int InferioresALimite(int LimSup, NoDaArvore No)
    {int Conta = 0;
     if (No == null)
      return Conta;
      else
       {if (No.PDado < LimSup)
        {return Conta +=1+
                InferioresALimite(LimSup,No.PEsquerda)+
                InferioresALimite(LimSup, No.PDireita);}
         else
            return Conta+=InferioresALimite(LimSup,No.PEsquerda);
    }}}
 public class ContagemNosInferiores
 {static void Main(string[] args)
  {Arvore T = new Arvore();
   int InteiroParaNo;
   Random R = new Random();
   Console.Write("Elementos extraídos aleatoriamente: ");
   for (int I = 1; I < 10; I++)
    {InteiroParaNo = R.Next(100);
     Console.Write(InteiroParaNo + " ");
     T.InserirNo(InteiroParaNo);}
   Console.Write("\nLimite superior para a contagem ");
   int LimSup = Convert.ToInt16(Console.ReadLine());
   int Conta = T.InferioresALimite(LimSup, T.PRaiz);
   Console.WriteLine("Número de elementos inferiores a {0} =
   {1}", LimSup, Conta);}}}
```

? EXERCÍCIO 23: ALTURA DE ÁRVORE BINÁRIA

```
using System;
namespace EL23
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public int PDado
   {get
    {return Dado;}
   set
```

```
    {Dado=value;}}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
   set
    {Esquerda=value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita=value;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (PEsquerda == null)
     PEsquerda = new NoDaArvore(InteiroParaNo);
     else
      PEsquerda.InserirRegisto(InteiroParaNo);}
      else if (InteiroParaNo > Dado)
            {if (PDireita == null)
             PDireita = new NoDaArvore(InteiroParaNo);
             else
              PDireita.InserirRegisto(InteiroParaNo);}}}

  public class Arvore
   {private NoDaArvore Raiz;
  public Arvore()
   {Raiz = null; }
  public NoDaArvore PRaiz
   {get
    {return Raiz;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}
  public int Altura(NoDaArvore C)
   {if (C == null)
    return 0;
    else
     return Maior(Altura(C.PEsquerda), Altura(C.PDireita)) + 1;}
  public int Maior(int X, int Y)
   {if (X >= Y)
    return X;
    else
     return Y;}}

public class AlturaDasArvores
 {static void Main(string[] args)
  {Arvore T = new Arvore();
```

```
    int InteiroParaNo;
    Random R = new Random();
    Console.Write("Os seguintes inteiros:");
    for (int I = 1; I < 10; I++)
     {InteiroParaNo = R.Next(100);
      Console.Write(InteiroParaNo + " ");
      T.InserirNo(InteiroParaNo);}
    Console.WriteLine("\nformam uma árvore com altura={0}",
    T.Altura(T.PRaiz));}}}
```

? EXERCÍCIO 24: REFLEXÃO DE ÁRVORE BINÁRIA

```
using System;
namespace EL24
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public int PDado
   {get
     {return Dado;}}
  public NoDaArvore PEsquerda
   {get
     {return Esquerda;}
    set
     {Esquerda=value;}}
  public NoDaArvore PDireita
   {get
     {return Direita;}
    set
     {Direita=value;}}
  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
      Esquerda.InserirRegisto(InteiroParaNo);}
    else if (InteiroParaNo > Dado)
           {if (Direita == null)
            Direita = new NoDaArvore(InteiroParaNo);
            else
             Direita.InserirRegisto(InteiroParaNo);}}}
```

```
public class Arvore
 {private NoDaArvore Raiz;
  public Arvore()
   {Raiz = null; }
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
    Raiz = new NoDaArvore(InteiroParaNo);
    else
     Raiz.InserirRegisto(InteiroParaNo);}
  public NoDaArvore PRaiz
   {get
    {return Raiz;}}
  public void Refletir(NoDaArvore No)
   {NoDaArvore Q;
   if (No != null)
    {Q = No.PEsquerda;
     No.PEsquerda = No.PDireita;
     No.PDireita = Q;
     Refletir(No.PEsquerda);
     Refletir(No.PDireita);}}
  public void PercursoPreOrdem()
   {PreOrdem(Raiz);}
  public void PreOrdem(NoDaArvore No)
   {if (No == null)
     return;
    Console.Write(No.PDado + " ");
    PreOrdem(No.PEsquerda);
    PreOrdem(No.PDireita);}}

public class ReflexaoDaArvore
 {static void Main(string[] args)
  {Arvore T = new Arvore();
   int InteiroParaNo;
   Random R = new Random();
   for (int I = 1; I < 10; I++)
    {InteiroParaNo = R.Next(100);
     T.InserirNo(InteiroParaNo);}
   Console.WriteLine("Árvore original:");
   T.PercursoPreOrdem();
   T.Refletir(T.PRaiz);
   Console.WriteLine("\nÁrvore refletida:");
   T.PercursoPreOrdem();
   Console.WriteLine();}}}
```

? EXERCÍCIO 25: PROCURA DE UM ELEMENTO DE UMA ÁRVORE BINÁRIA

```
using System;
namespace EL25
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private int Dado;
  public NoDaArvore(int D)
   {Dado = D;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
   set
    {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita = value;}}
  public int PDado
   {get
    {return Dado;}
    set
    {Dado = value;}}

  public void InserirRegisto(int InteiroParaNo)
   {if (InteiroParaNo < Dado)
    {if (Esquerda == null)
     Esquerda = new NoDaArvore(InteiroParaNo);
     else
      Esquerda.InserirRegisto(InteiroParaNo);}
      else if (InteiroParaNo > Dado)
            {if (Direita == null)
             Direita = new NoDaArvore(InteiroParaNo);
             else
              Direita.InserirRegisto(InteiroParaNo);}}}

 public class Arvore
  {private NoDaArvore Raiz;
   public Arvore()
   {Raiz = null; }
  public NoDaArvore PRaiz
   {get
    {return Raiz; }
   set
```

```
    {Raiz = value;}}
  public void InserirNo(int InteiroParaNo)
   {if (Raiz == null)
      Raiz = new NoDaArvore(InteiroParaNo);
      else
       Raiz.InserirRegisto(InteiroParaNo);}
  public void PercursoPreOrdem()
   {PreOrdem(Raiz);}
  public void PreOrdem(NoDaArvore No)
   {if (No == null)
    return;
    Console.Write(No.PDado + " ");
    PreOrdem(No.PEsquerda);
    PreOrdem(No.PDireita);}
  public bool existe(NoDaArvore Q, int X)
   {if (Q == null)
    return false;
    else
    {if (Q.PDado == X)
     return true;
     else
      {if (Q.PDado > X)
       return existe(Q.PEsquerda, X);
       else
        return existe(Q.PDireita, X);}}}}

 public class ProcuraElemento
  {static void Main(string[] args)
   {Arvore T = new Arvore();
    int InteiroParaNo;
    Random R = new Random();
    for (int I = 1; I < 10; I++)
     {InteiroParaNo = R.Next(100);
      T.InserirNo(InteiroParaNo);}
    T.PercursoPreOrdem();
    Console.Write("\nQue número procura? ");
    int X=Convert.ToInt16(Console.ReadLine());
    Console.WriteLine(T.existe(T.PRaiz, X)? X+ " existe":X+" não
    existe");}}}
```

? EXERCÍCIO 26: ÁRVORE BINÁRIA POR ORDEM ALFABÉTICA

```
using System;
namespace EL26
{public class NoDaArvore
 {private NoDaArvore Esquerda;
```

```
  private NoDaArvore Direita;
  private string Pal;
  public NoDaArvore(string Palavra)
   {Pal = Palavra;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
    set
    {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
    set
    {Direita = value;}}
  public string PPalavra
   {get
    {return Pal;}
    set
    {Pal = value;}}
  public void InserirRegisto(string Palavra)
   {if (Palavra.CompareTo(Pal)<0)
    {if (PEsquerda == null)
     PEsquerda = new NoDaArvore(Palavra);
     else
      PEsquerda.InserirRegisto(Palavra);}
      else if (Palavra.CompareTo(Pal) > 0)
             {if (PDireita == null)
              PDireita = new NoDaArvore(Palavra);
              else
               PDireita.InserirRegisto(Palavra);}}}

public class ArvAlfa
 {private NoDaArvore Raiz;
 public ArvAlfa()
  {Raiz = null;}
 public NoDaArvore PRaiz
  {get
   {return Raiz;}
  set
   {Raiz = value;}}
 public void InserirNo(string Palavra)
   {if (Raiz == null)
    Raiz = new NoDaArvore(Palavra);
    else
     Raiz.InserirRegisto(Palavra);}
 public void PercursoEmOrdem()
  {EmOrdem(Raiz);}
 public void EmOrdem(NoDaArvore No)
```

```
   {if (No == null)
    return;
    EmOrdem(No.PEsquerda);
    Console.Write(No.PPalavra + " ");
    EmOrdem(No.PDireita);}}

 public class ArvoreOrdemAlfabetica
  {static void Main(string[] args)
   {string[] Palavras={"Mesa","Cadeira","Afiador","Diário",
    "Caneta", "Lápis", "Caderno", "Régua"};
    ArvAlfa T = new ArvAlfa();
    for (int I = 0; I <= Palavras.Length-1; I++)
     T.InserirNo(Palavras[I]);
    T.PercursoEmOrdem();
    Console.WriteLine();}}}
```

? EXERCÍCIO 27: ELIMINAÇÃO DA SUBÁRVORE DA ESQUERDA

```
using System;
namespace EL27
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private string Pal;
  public NoDaArvore(string Palavra)
   {Pal = Palavra;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
    set
    {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita = value;}}
  public string PPalavra
   {get
    {return Pal;}
   set
    {Pal = value;}}
  public void InserirRegisto(string Palavra)
   {if (Palavra.CompareTo(Pal) < 0)
    {if (PEsquerda == null)
     PEsquerda = new NoDaArvore(Palavra);
     else
```

```
        PEsquerda.InserirRegisto(Palavra);}
        else if (Palavra.CompareTo(Pal) > 0)
              {if (PDireita == null)
               PDireita = new NoDaArvore(Palavra);
               else
                PDireita.InserirRegisto(Palavra);}}}

public class ArvAlfa
 {private NoDaArvore Raiz;

  public ArvAlfa()
   {Raiz = null;}
  public void InserirNo(string Palavra)
   {if (Raiz == null)
    Raiz = new NoDaArvore(Palavra);
    else
     Raiz.InserirRegisto(Palavra);}
  public NoDaArvore PRaiz
   {get
    {return Raiz;}
   set
   {Raiz = value;}}
  public void PercursoEmOrdem()
   {EmOrdem(Raiz);}
  public void EmOrdem(NoDaArvore No)
   {if (No == null)
    return;
   EmOrdem(No.PEsquerda);
   Console.Write(No.PPalavra + " ");
   EmOrdem(No.PDireita);}
  public void EliminarSubArvore(String S, NoDaArvore C)
   {if (C != null)
    {if (C.PPalavra.CompareTo(S) == 0)
     C.PEsquerda = null;
      else
       {if (C.PPalavra.CompareTo(S) > 0)
        C = C.PEsquerda;
        else
         C = C.PDireita;
         EliminarSubArvore(S, C);}}}}

public class EliminacaoDeSubArvore
 {static void Main(string[] args)
  {string[] Palavras={"Mesa","Cadeira","Afiador","Diário",
   "Caneta","Lápis","Caderno","Régua","Caixa"};
   ArvAlfa T = new ArvAlfa();
   for (int I = 0; I <= Palavras.Length - 1; I++)
    T.InserirNo(Palavras[I]);
   Imprimir(T, "Árvore original:");
```

```
    Console.Write("Que palavra procura? ");
    string Pal=Console.ReadLine();
    T.EliminarSubArvore(Pal, T.PRaiz);
   Imprimir(T, "Árvore depois da eliminação:");}
  static void Imprimir(ArvAlfa T, string Tit)
  {Console.WriteLine(Tit);
   T.PercursoEmOrdem();
   Console.WriteLine();}}}
```

? EXERCÍCIO 28: ELIMINAÇÃO DE UMA FOLHA DE UMA ÁRVORE BINÁRIA

```
using System;
namespace EL28
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private string Pal;
  public NoDaArvore(string Palavra)
   {Pal = Palavra;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
   set
    {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita = value;}}
  public string PPalavra
   {get
    {return Pal;}
   set
    {Pal = value;}}
  public void InserirRegisto(string Palavra)
   {if (Palavra.CompareTo(Pal) < 0)
    {if (PEsquerda == null)
     PEsquerda = new NoDaArvore(Palavra);
     else
      PEsquerda.InserirRegisto(Palavra);}
       else if (Palavra.CompareTo(Pal) > 0)
             {if (PDireita == null)
              PDireita = new NoDaArvore(Palavra);
              else
               PDireita.InserirRegisto(Palavra);}}}
```

```
public class ArvAlfa
 {private NoDaArvore Raiz;
  public ArvAlfa()
   {Raiz = null; }
  public void InserirNo(string Palavra)
   {if (Raiz == null)
    Raiz = new NoDaArvore(Palavra);
    else
     Raiz.InserirRegisto(Palavra);}
  public NoDaArvore PRaiz
   {get
    {return Raiz;}
   set
    {Raiz = value;}}
  public void PercursoEmOrdem()
   {EmOrdem(Raiz); }
  public void EmOrdem(NoDaArvore No)
   {if (No == null)
    return;
    EmOrdem(No.PEsquerda);
    Console.Write(No.PPalavra + " ");
    EmOrdem(No.PDireita);}

  public void EliminarFolha(String S)
   {NoDaArvore Corrente=Raiz;
    NoDaArvore Anterior=Corrente;
    string Direcao = "Esquerda";
    Eliminar(S, Corrente, Anterior, Direcao);}
  public void Eliminar (string S, NoDaArvore C, NoDaArvore A,
  string Direcao)
   {if (C != null)
    {if (C.PPalavra.CompareTo(S) == 0)
     {if (C.PEsquerda == null && C.PDireita == null)
      {if (Direcao == "Esquerda")
       A.PEsquerda = null;
       else
        A.PDireita = null;}
        else
         Console.WriteLine("{0} não é uma folha da árvore,
         portanto, não eliminamos!!!", S);}
         else
          {A = C;
          if (C.PPalavra.CompareTo(S) > 0)
           {Direcao = "Esquerda";
            C = A.PEsquerda;}
            else
             {Direcao = "Direita";
              C = A.PDireita;}
             Eliminar(S, C , A, Direcao);}}}}
```

```
public class EliminacaoDeFolha
 {static void Main(string[] args)
  {string[]
  Palavras={"Mesa","Cadeira","Afiador","Diário","Caneta",
  "Lápis", "Caderno", "Régua" };
   ArvAlfa T = new ArvAlfa();
   for (int I = 0; I <= Palavras.Length - 1; I++)
   T.InserirNo(Palavras[I]);
   Imprimir(T, "Árvore original:");
   Console.Write("Que palavra procura? ");
   string Pal=Console.ReadLine();
   T.EliminarFolha(Pal);
   Imprimir(T, "Árvore depois da eliminação:");}
  static void Imprimir(ArvAlfa T, string Tit)
   {Console.WriteLine(Tit);
    T.PercursoEmOrdem();
    Console.WriteLine();}}}
```

? EXERCÍCIO 29: ELIMINAÇÃO DA RAIZ DE UMA ÁRVORE BINÁRIA

```
using System;
namespace EL29
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private string Pal;
  public NoDaArvore(string Palavra)
   {Pal = Palavra;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
   set
    {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita = value;}}
  public string PPalavra
   {get
    {return Pal;}
   set
    {Pal = value;}}
```

```
  public void InserirRegisto(string Palavra)
   {if (Palavra.CompareTo(Pal) < 0)
    {if (PEsquerda == null)
     PEsquerda = new NoDaArvore(Palavra);
     else
      PEsquerda.InserirRegisto(Palavra);}
     else if (Palavra.CompareTo(Pal) > 0)
      {if (PDireita == null)
        PDireita = new NoDaArvore(Palavra);
        else
        PDireita.InserirRegisto(Palavra);}}}

public class ArvAlfa
 {private NoDaArvore Raiz;
  public ArvAlfa()
   {Raiz = null;}
  public void InserirNo(string Palavra)
   {if (Raiz == null)
    Raiz = new NoDaArvore(Palavra);
    else
     Raiz.InserirRegisto(Palavra);}
  public NoDaArvore PRaiz
   {get
    {return Raiz;}
   set
    {Raiz = value;}}
  public void PercursoEmOrdem()
   {EmOrdem(Raiz);}
  public void EmOrdem(NoDaArvore No)
   {if (No == null)
    return;
    EmOrdem(No.PEsquerda);
    Console.Write(No.PPalavra + " ");
    EmOrdem(No.PDireita);}
  public void EliminarRaiz()
   {NoDaArvore SubArvoreEsq = PRaiz.PEsquerda;
    NoDaArvore SubArvoreDir = PRaiz.PDireita;
    NoDaArvore SubArvoreEsqDir;
    NoDaArvore Anterior;
    if (SubArvoreEsq != null)
     {PRaiz = SubArvoreEsq;
      Anterior = SubArvoreEsq;
      SubArvoreEsqDir = SubArvoreEsq.PDireita;
      while (SubArvoreEsqDir != null)
       {Anterior = SubArvoreEsqDir;
        SubArvoreEsqDir = SubArvoreEsqDir.PDireita;}
        Anterior.PDireita = SubArvoreDir;}
    else
    {PRaiz = SubArvoreDir;}}}
```

```
public class EliminacaoRaiz
 {static void Main(string[] args)
  {string[] Palavras ={"Mesa","Cadeira","Afiador","Diário",
   "Caneta", "Caderno", "Canudo", "Caixa","Lápis", "Régua"};
   ArvAlfa T = new ArvAlfa();
   for (int I = 0; I <= Palavras.Length - 1; I++)
    T.InserirNo(Palavras[I]);
   Imprimir(T, "Árvore original:");
   T.EliminarRaiz();
   Imprimir(T, "Árvore com nova raiz:");}
  static void Imprimir(ArvAlfa T, string Tit)
   {Console.WriteLine(Tit);
    T.PercursoEmOrdem();
    Console.WriteLine();}}}
```

? EXERCÍCIO 30: ELIMINAÇÃO DE UM ELEMENTO DE UMA ÁRVORE BINÁRIA

```
using System;
namespace EL30
{public class NoDaArvore
 {private NoDaArvore Esquerda;
  private NoDaArvore Direita;
  private string Pal;
  public NoDaArvore(string Palavra)
   {Pal = Palavra;
    Esquerda = Direita = null;}
  public NoDaArvore PEsquerda
   {get
    {return Esquerda;}
   set
   {Esquerda = value;}}
  public NoDaArvore PDireita
   {get
    {return Direita;}
   set
    {Direita = value;}}
  public string PPalavra
   {get
    {return Pal;}
   set
    {Pal = value;}}
 public void InserirRegisto(string Palavra)
  {if (Palavra.CompareTo(Pal) < 0)
   {if (PEsquerda == null)
    PEsquerda = new NoDaArvore(Palavra);
    else
```

```
     PEsquerda.InserirRegisto(Palavra);}
     else if (Palavra.CompareTo(Pal) > 0)
           {if (PDireita == null)
            PDireita = new NoDaArvore(Palavra);
            else
            PDireita.InserirRegisto(Palavra);}}}

public class ArvAlfa
 {private NoDaArvore Raiz;
  public ArvAlfa()
   {Raiz = null;}
  public NoDaArvore PRaiz
   {get
    {return Raiz;}
   set
    {Raiz = value;}}
  public void InserirNo(string Palavra)

   {if (Raiz == null)
    Raiz = new NoDaArvore(Palavra);
    else
     Raiz.InserirRegisto(Palavra);}
  public void PercursoEmOrdem()
   {EmOrdem(Raiz);}
 public void EmOrdem(NoDaArvore No)
   {if (No == null)
    return;
    EmOrdem(No.PEsquerda);
    Console.Write(No.PPalavra + " ");
    EmOrdem(No.PDireita);}
public void EliminarElemento(String S)
 {NoDaArvore Corrente = Raiz;
  NoDaArvore Anterior = Raiz;
  string Direcao = "Esquerda";
  Procurar(S, Corrente, Anterior, Direcao);}
public void Procurar(string S, NoDaArvore C, NoDaArvore A,
string Direcao)
 {if (C != null)
  {if (C.PPalavra.CompareTo(S) == 0)
   EliminarEle(C, A, Direcao);
   else
    {A = C;
    if (C.PPalavra.CompareTo(S) > 0)
    {Direcao = "Esquerda";
    C = C.PEsquerda;}
    else
    {Direcao = "Direita";
    C = C.PDireita;}
    Procurar(S, C, A, Direcao);}}}
```

```
public void EliminarEle(NoDaArvore C, NoDaArvore A, string
Direcao)
  {if (C.PEsquerda == null || C.PDireita == null)
     Console.WriteLine("Não se aplica aqui");
     else
     {if (Direcao == "Esquerda")
      A.PEsquerda = C.PEsquerda;
      else
       A.PDireita = C.PEsquerda;
       NoDaArvore Descer = C.PEsquerda;
       NoDaArvore SubDireita = C.PDireita;
       NoDaArvore AntesDeDescer=Descer;
       while (Descer != null)
        {AntesDeDescer = Descer;
         Descer = Descer.PDireita;}
       AntesDeDescer.PDireita = SubDireita;}}}

public class EliminacaoDeElemento
 {static void Main(string[] args)
  {string[] Palavras ={"Mesa","Cadeira","Afiador","Diário",
   "Caneta", "Caderno", "Canudo", "Caixa","Lápis", "Régua"};
   ArvAlfa T = new ArvAlfa();
   for (int I = 0; I <= Palavras.Length - 1; I++)
    T.InserirNo(Palavras[I]);
   Imprimir(T, "Árvore original:");
   Console.Write("Que palavra procura? ");
   string Pal=Console.ReadLine();
   T.EliminarElemento(Pal);
   Imprimir(T, "Árvore depois da eliminação:");}
 static void Imprimir(ArvAlfa T, string Tit)
  {Console.WriteLine(Tit);
   T.PercursoEmOrdem();
   Console.WriteLine();}}}
```