Ex1:

```
using System;
using System.IO;
using System.Text;

public class Node
{
   private char item;
   private Node next;

   public Node(char item)
   {
      this.item = item;
      this.next = null;
   }

   public Node()
   {
      this.item = ' ';
      this.next = null;
   }

   public Node Next
   {
      get { return next; }
      set { next = value; }
   }

   public char Item
   {
      get { return item; }
      set { item = value; }
   }
}

class Pilha
{
   private Node top;

   public Pilha()
   {
      top = null;
   }

   public bool Vazio()
   {
      return top == null;
```

```
    }

    public void Empilhar(char x)
    {
        Node tmp = new Node(x);
        tmp.Next = top;
        top = tmp;
        tmp = null;
    }

    public char Desempilhar()
    {
        if (Vazio())
        {
            return ' ';
        }

        char item = top.Item;
        top = top.Next;
        return item;
    }

    public bool Verificar(char x)
    {
        char tmp = Desempilhar();
        if (x == ')' && tmp == '(')
        {
            return true;
        }
        else if ((x == ']' && tmp == '['))
        {
            return true;
        }
        else if ((x == '[' || x == '('))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}

class Program
{
    public static void Main(string[] args)
    {
```

```
    Console.WriteLine("Informe a sequencia:");
    string seq = Console.ReadLine().Replace(" ", "");
    Pilha teste = new Pilha();
    bool ok = true;
    for (int i = 0; i < seq.Length; i++)
    {
        if (seq[0] == ']' || seq[0] == ')')
        {
            ok = false;
            break;
        }
        else
        {
            if (i == 0)
            {
                teste.Empilhar(seq[i]);
                continue;
            }
            else
            {
                if (seq[i] == '[' || seq[i] == '(')
                {
                    teste.Empilhar(seq[i]);
                }
                else
                {
                    ok = teste.Verificar(seq[i]);
                    if (!ok)
                    {
                        break;
                    }
                }
            }
        }
    }

    if (ok && teste.Vazio())
    {
        Console.Write("Sequencia bem formada!");
    }
    else
    {
        Console.Write("Sequencia mal formada!");
    }
  }
}
```

Ex2:

```
using System;

namespace CSharpLista7
{
   public class Node
   {
      private int numb;
      private Node next;

      public Node(int numb)
      {
         this.numb = numb;
         this.next = null;
      }

      public Node()
      {
         this.numb = 0;
         this.next = null;
      }

      public Node Next
      {
         get { return next; }
         set { next = value; }
      }

      public int Item
      {
         get { return numb; }
         set { numb = value; }
      }
   }

   public class Octal
   {
      private Node top;

      public Octal()
      {
         top = null;
      }

      public void Empilhar(int x)
```

```
        {
            Node tmp = new Node(x);
            tmp.Next = top;
            top = tmp;
            tmp = null;
        }

        public string Mostrar(Octal obj)
        {
            string ret = "";
            for (Node i = obj.top; i != null; i = i.Next)
            {
                ret += i.Item.ToString();
            }

            return ret;
        }

        public static void Main()
        {
            Console.WriteLine("Informe um numero:");
            int num = int.Parse(Console.ReadLine().Replace(" ", ""));
            Octal teste = new Octal();
            do
            {
                teste.Empilhar(num % 8);
                num /= 8;
            } while (num > 0);

            Console.Write($"Octal: {teste.Mostrar(teste)}");
        }
    }
}
```

Ex3:

```
using System;

namespace CSharpLista7
{
    public class Node
    {
        private string arq;
        private int qtd;
        private Node next;

        public Node(string arq, int qtd)
        {
```

```
        this.arq = arq;
        this.qtd = qtd;
        this.next = null;
    }

    public Node()
    {
        this.arq = "";
        this.qtd = 0;
        this.next = null;
    }

    public Node Next
    {
        get { return next; }
        set { next = value; }
    }

    public string Arq
    {
        get { return arq; }
        set { arq = value; }
    }

    public int Qtd
    {
        get { return qtd; }
        set { qtd = value; }
    }
}

public class Fila
{
    private Node init, end;

    public Fila()
    {
        init = new Node();
        end = init;
    }

    public void Inserir()
    {
        Console.WriteLine("Digite o nome do arquivo:");
        string x = Console.ReadLine();
        Console.WriteLine("Digite a quantidade de paginas:");
        int y = Convert.ToInt32(Console.ReadLine());
        end.Next = new Node(x, y);
```

```
    end = end.Next;
}

public (string, int) Imprimir()
{
    if (init == end)
    {
        throw new Exception("Fila de impressao esta vazia");
    }

    Node tmp = init;
    init = init.Next;
    string _aqr = init.Arq;
    int _qtd = init.Qtd;
    tmp.Next = null;
    tmp = null;
    return (_aqr, _qtd);
}
public void Mostrar()
{
    for (Node i = init.Next; i.Next != null; i = i.Next)
    {
        Console.WriteLine($"nome: {i.Arq}, numero paginas: {i.Qtd}");
    }
    Console.WriteLine($"nome: {end.Arq}, numero paginas: {end.Qtd}");
}

public static void Main()
{
    Fila teste = new Fila();

    bool continuar = true;

    while (continuar)
    {
        Console.WriteLine("1. Inserir arquivo na fila de impressao");
        Console.WriteLine("2. Executar impressao");
        Console.WriteLine("3. Exibir fila de impressao");
        Console.WriteLine("4. Sair");
        int opcao = Convert.ToInt32(Console.ReadLine());

        switch (opcao)
        {
            case 1:
                teste.Inserir();
                break;
            case 2:
                teste.Imprimir();
```

```
                break;
            case 3:
                teste.Mostrar();
                break;
            case 4:
                Console.WriteLine("O programa sera encerrado.");
                continuar = false;
                break;
            default:
                Console.WriteLine("Opção inválida.");
                break;
        }
      }
    }
  }
}
```

Ex4:

```
using System;

namespace CSharpLista7
{
  public class Node
  {
    private string aluno;
    private Node next;

    public Node(string aluno)
    {
      this.aluno = aluno;
      this.next = null;
    }

    public Node()
    {
      this.aluno = "";
      this.next = null;
    }

    public Node Next
    {
      get { return next; }
      set { next = value; }
    }

    public string Aluno
```

```
    {
        get { return aluno; }
        set { aluno = value; }
    }
}

public class Aluno
{

    private Node init, end;

    public Aluno()
    {
        init = new Node();
        end = init;
    }

    public void Inserir()
    {
        Console.WriteLine("Informe o nome do aluno:");
        string aluno = Console.ReadLine();
        end.Next = new Node(aluno);
        end = end.Next;
    }

    public string Remover()
    {
        if (init.Next == null)
        {
            throw new Exception("Lista de alunos vazia");
        }
        Node tmp = init.Next;
        if (tmp == end)
        {
            end = init;
        }
        init.Next = tmp.Next;
        string ret = tmp.Aluno;
        return ret;
    }

    public void Mostrar()
    {
        // Console.WriteLine(init.Aluno);
        for (Node i = init; i.Next != null; i = i.Next)
        {
            Console.WriteLine(i.Aluno);
        }
```

```
    Console.WriteLine(end.Aluno);
}

public bool Pesquisar()
{
    Console.WriteLine("Informe o nome do aluno:");
    string tmp = Console.ReadLine();
    bool encontrado = tmp == end.Aluno;
    for (Node i = init; !encontrado && i.Next != null; i = i.Next)
    {
        if (i.Aluno == tmp)
        {
            return true;
        }
    }
    return encontrado;
}

public static void Main()
 {
     Aluno ic = new Aluno();
     Aluno mestrado = new Aluno();
     bool continuar = true;

     while (continuar)
     {
         Console.WriteLine("Menu:");
         Console.WriteLine("1. Inserir um aluno na fila de espera de bolsas de IC");
         Console.WriteLine("2. Inserir um aluno na fila de espera de bolsas de Mestrado");
         Console.WriteLine("3. Remover um aluno da fila de bolsas de IC");
         Console.WriteLine("4. Remover um aluno da fila de bolsas de Mestrado");
         Console.WriteLine("5. Mostrar fila de espera de bolsas de IC");
         Console.WriteLine("6. Mostrar fila de espera de bolsas de Mestrado");
         Console.WriteLine("7. Pesquisar aluno na fila de espera de bolsas de IC");
         Console.WriteLine("8. Pesquisar aluno na fila de espera de bolsas de Mestrado");
         Console.WriteLine("9. Sair");
         int opcao = Convert.ToInt32(Console.ReadLine());

         switch (opcao)
         {
             case 1:
                 ic.Inserir();
                 break;
             case 2:
                 mestrado.Inserir();
                 break;
             case 3:
                 Console.WriteLine($"Aluno removido: {ic.Remover()}");
```

```
                break;
            case 4:
                Console.WriteLine($"Aluno removido: {mestrado.Remover()}");
                break;
            case 5:
                Console.Write("Fila de Espera IC:");
                ic.Mostrar();
                break;
            case 6:
                Console.Write("Fila de Espera Mestrado:");
                mestrado.Mostrar();
                break;
            case 7:
                bool okIc = ic.Pesquisar();
                if (okIc)
                {
                    Console.WriteLine("Aluno ja consta na fila de IC");
                }else
                {
                    Console.WriteLine("Aluno nao consta na fila de IC");
                }
                break;
            case 8:
                bool okM = mestrado.Pesquisar();
                if (okM)
                {
                    Console.WriteLine("Aluno ja consta na fila de Mestrado");
                }
                else
                {
                    Console.WriteLine("Aluno nao consta na fila de Mestrado");
                }
                break;
            case 9:
                Console.Write("O programa sera encerrado");
                continuar = false;
                break;
            default:
                Console.WriteLine("Opção inválida.");
                break;
        }
    }
}

}
}
```

Ex5:

```
using System;

namespace CSharpLista7
{
   public class DobleNode
   {
      private int tempo;
      private DobleNode next;
      private DobleNode previous;

      public DobleNode(int tempo)
      {
         this.tempo = tempo;
         this.next = null;
         this.previous = null;
      }

      public DobleNode()
      {
         this.tempo = 0;
         this.next = null;
         this.previous = null;
      }

      public DobleNode Next
      {
         get { return next; }
         set { next = value; }
      }

      public DobleNode Previeous
      {
         get { return previous; }
         set { previous = value; }
      }

      public int Tempo
      {
         get { return tempo; }
         set { tempo = value; }
      }
   }

   public class Corredor
   {
      private DobleNode init, end;
```

```
public Corredor()
{
    init = new DobleNode();
    end = init;
}


public void InserirInit(int x)
{
    DobleNode tmp = new DobleNode(x);
    tmp.Previeous = init;
    tmp.Next = init.Next;
    //
    if (init.Next != null)
    {
        init.Next.Previeous = tmp;
    }
    else
    {
        end = tmp;
    }

    init.Next = tmp;
}


public void InserirEnd(int x)
{
    DobleNode tmp = new DobleNode(x);
    tmp.Previeous = end;
    end.Next = tmp;
    end = tmp;
}


public void Inserir(int pos, int x)
{
    if (pos < 0)
    {
        Console.WriteLine("Posição inválida");
        return;
    }

    DobleNode tmp = new DobleNode(x);
    DobleNode current = init;
    int currentIndex = 0;
```

```
    while (current.Next != null && currentIndex < pos)
    {
        current = current.Next;
        currentIndex++;
    }

    if (currentIndex == pos)
    {
        tmp.Next = current.Next;
        if (current.Next != null)
        {
            current.Next.Previeous = tmp;
        }

        tmp.Previeous = current;
        current.Next = tmp;

        if (tmp.Next == null)
        {
            end = tmp;
        }
    }
    else
    {
        Console.WriteLine("Posição maior que o tamanho da lista");
    }
}

public void Mostrar()
{
    Console.Write("[");
    DobleNode current = init.Next;
    while (current != null)
    {
        Console.Write($"{current.Tempo} ");
        current = current.Next;
    }

    Console.WriteLine("]");
}

public int RemoverInit()
{
    if (init == end)
    {
        throw new Exception("Lista vazia");
    }
```

```
    DobleNode rem = init.Next;
    int tmp = rem.Tempo;
    init.Next = rem.Next;
    if (rem.Next != null)
    {
        rem.Next.Previeous = init;
    }

    rem = null;
    return tmp;
}


public int RemoverEnd()
{
    if (init == end)
    {
        throw new Exception("Lista vazia");
    }

    DobleNode rem = end;
    int tmp = rem.Tempo;
    end = end.Previeous;
    if (end != null)
    {
        end.Next = null;
    }
    else
    {
        init.Next = null;
    }

    rem = null;
    return tmp;
}


public int Remover(int pos)
{
    if (init.Next == null)
    {
        throw new Exception("Lista vazia");
    }

    if (pos == 0)
    {
        return RemoverInit();
    }
```

```
    DobleNode current = init.Next;
    int count = 0;

    while (current != null && count < pos)
    {
        if (count + 1 == pos)
        {
            if (current.Next == end)
            {
                return RemoverEnd();
            }

            int temp = current.Next.Tempo;
            current.Next = current.Next.Next;
            if (current.Next != null)
            {
                current.Next.Previeous = current;
            }

            return temp;
        }

        current = current.Next;
        count++;
    }

    throw new Exception("Posição não encontrada");
}


public void RemoverItem(int x)
{
    if (init.Next == null)
    {
        Console.WriteLine("Lista vazia");
        return;
    }

    int count = 0;
    for (DobleNode i = init.Next; i != null; i = i.Next, count++)
    {
        if (i.Tempo == x)
        {
            Remover(count);
            return;
        }
    }
```

```
    Console.WriteLine("Item não encontrado");
}

public int CountItens(int x)
{
    int count = 0;
    for (DobleNode i = init.Next; i != null; i = i.Next)
    {
        if (i.Tempo == x)
        {
            count++;
        }
    }

    return count;
}

public static void Main()
{
    Corredor teste = new Corredor();
    bool continuar = true;

    while (continuar)
    {
        Console.WriteLine("Menu:");
        Console.WriteLine("1) Inserir um tempo no inicio da lista.");
        Console.WriteLine("2) Inserir um tempo no final da lista.");
        Console.WriteLine("3) Inserir um tempo numa posicao especifica da lista.");
        Console.WriteLine("4) Remover o primeiro tempo da lista.");
        Console.WriteLine("5) Remover o ultimo tempo da lista.");
        Console.WriteLine("6) Remover um tempo de uma posicao especifica na lista.");
        Console.WriteLine("7) Remover um tempo especifico da lista.");
        Console.WriteLine("8) Pesquisar quantas vezes um determinado tempo consta na lista.");
        Console.WriteLine("9) Mostrar todos os tempos da lista.");
        Console.WriteLine("10) Encerrar o programa.");
        int input = Convert.ToInt32(Console.ReadLine());

        switch (input)
        {
            case 1:
                Console.WriteLine("Informe o tempo:");
                int tempoInit = Convert.ToInt32(Console.ReadLine());
                teste.InserirInit(tempoInit);
                break;
            case 2:
                Console.WriteLine("Informe o tempo:");
                int tempoEnd = Convert.ToInt32(Console.ReadLine());
```

```
                teste.InserirEnd(tempoEnd);
                break;
            case 3:
                Console.WriteLine("Informe o tempo:");
                int tempoPos = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine("Informe a posicao:");
                int pos = Convert.ToInt32(Console.ReadLine());
                teste.Inserir(pos, tempoPos);
                break;
            case 4:
                Console.WriteLine($"Tempo removido: {teste.RemoverInit()}.");
                break;
            case 5:
                Console.WriteLine($"Tempo removido: {teste.RemoverEnd()}.");
                break;
            case 6:
                Console.WriteLine("Informe a posicao:");
                int posRemove = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine($"Tempo removido: {teste.Remover(posRemove)}.");
                break;
            case 7:
                Console.WriteLine("Informe o tempo a remover:");
                int tempoRemove = Convert.ToInt32(Console.ReadLine());
                teste.RemoverItem(tempoRemove);
                break;
            case 8:
                Console.WriteLine("Informe o tempo:");
                int tempoCount = Convert.ToInt32(Console.ReadLine());
                Console.WriteLine($"Quantidade: {teste.CountItens(tempoCount)}");
                break;
            case 9:
                teste.Mostrar();
                break;
            default:
                Console.Write("O programa sera encerrado.");
                continuar = false;
                break;
        }


    }
  }
 }
}
```

Ex6:

```
using System;

namespace CSharpLista7
{
   public class Site
   {
      public string Nome { get; set; }
      public string Link { get; set; }

      public Site(string nome, string link)
      {
         Nome = nome;
         Link = link;
      }
   }

   public class Celula
   {
      public Site Elemento { get; set; }
      public Celula Proximo { get; set; }

      public Celula(Site elemento)
      {
         Elemento = elemento;
         Proximo = null;
      }
   }

   public class Lista
   {
      private Celula inicio;

      public Lista()
      {
         inicio = null;
      }

      public void InserirInicio(Site site)
      {
         Celula nova = new Celula(site) { Proximo = inicio };
         inicio = nova;
      }

      public void InserirFim(Site site)
      {
```

```
    Celula nova = new Celula(site);
    if (inicio == null)
    {
        inicio = nova;
    }
    else
    {
        Celula atual = inicio;
        while (atual.Proximo != null)
        {
            atual = atual.Proximo;
        }

        atual.Proximo = nova;
    }
}

public void InserirPosicao(Site site, int posicao)
{
    Celula nova = new Celula(site);
    if (posicao == 0)
    {
        nova.Proximo = inicio;
        inicio = nova;
    }
    else
    {
        Celula atual = inicio;
        for (int i = 0; i < posicao - 1 && atual != null; i++)
        {
            atual = atual.Proximo;
        }

        if (atual != null)
        {
            nova.Proximo = atual.Proximo;
            atual.Proximo = nova;
        }
    }
}

public string RemoverInicio()
{
    if (inicio == null)
    {
        throw new Exception("Lista vazia");
    }
```

```
    string nome = inicio.Elemento.Nome;
    inicio = inicio.Proximo;
    return nome;
}

public string RemoverFim()
{
    if (inicio == null)
    {
        throw new Exception("Lista vazia");
    }

    if (inicio.Proximo == null)
    {
        string nome = inicio.Elemento.Nome;
        inicio = null;
        return nome;
    }

    Celula atual = inicio;
    while (atual.Proximo.Proximo != null)
    {
        atual = atual.Proximo;
    }

    string nomeRemovido = atual.Proximo.Elemento.Nome;
    atual.Proximo = null;
    return nomeRemovido;
}

public string RemoverPosicao(int posicao)
{
    if (inicio == null)
    {
        throw new Exception("Lista vazia");
    }

    if (posicao == 0)
    {
        return RemoverInicio();
    }

    Celula atual = inicio;
    Celula anterior = null;
    for (int i = 0; i < posicao && atual != null; i++)
    {
        anterior = atual;
        atual = atual.Proximo;
```

```
    }

    if (atual == null)
    {
        throw new Exception("Posição inválida");
    }

    anterior.Proximo = atual.Proximo;
    return atual.Elemento.Nome;
}

public void Mostrar()
{
    Celula atual = inicio;
    while (atual != null)
    {
        Console.WriteLine($"{atual.Elemento.Nome}: {atual.Elemento.Link}");
        atual = atual.Proximo;
    }
}

public string PesquisarLink(string nomeSite)
{
    Celula atual = inicio;
    while (atual != null)
    {
        if (atual.Elemento.Nome.Equals(nomeSite, StringComparison.OrdinalIgnoreCase))
        {
            return atual.Elemento.Link;
        }

        atual = atual.Proximo;
    }

    return "Site não encontrado";
}

public static void Main()
{
    Lista teste = new Lista();
    bool continuar = true;

    while (continuar)
    {
        Console.WriteLine("Menu:");
        Console.WriteLine("1) Inserir um Site no inicio da lista");
        Console.WriteLine("2) Inserir um Site no final da lista");
        Console.WriteLine("3) Inserir um Site numa posicao especifica da lista");
```

```
Console.WriteLine("4) Remover o primeiro Site da lista");
Console.WriteLine("5) Remover o ultimo Site da lista");
Console.WriteLine("6) Remover um Site de uma posicao especifica da lista");
Console.WriteLine("7) Mostrar o nome e o link de todos os sites da lista");
Console.WriteLine("8) Pesquisar o link de um site");
Console.WriteLine("9) Encerrar o programa");

int option = Convert.ToInt32(Console.ReadLine());

switch (option)
{
    case 1:
        Console.WriteLine("Informe o nome do site:");
        string nome1 = Console.ReadLine();
        Console.WriteLine("Informe o link do site:");
        string link1 = Console.ReadLine();
        teste.InserirInicio(new Site(nome1, link1));
        break;
    case 2:
        Console.WriteLine("Informe o nome do site:");
        string nome2 = Console.ReadLine();
        Console.WriteLine("Informe o link do site:");
        string link2 = Console.ReadLine();
        teste.InserirFim(new Site(nome2, link2));
        break;
    case 3:
        Console.WriteLine("Informe o nome do site:");
        string nome3 = Console.ReadLine();
        Console.WriteLine("Informe o link do site:");
        string link3 = Console.ReadLine();
        Console.WriteLine("Informe a posicao:");
        int posicao3 = Convert.ToInt32(Console.ReadLine());
        teste.InserirPosicao(new Site(nome3, link3), posicao3);
        break;
    case 4:
        try
        {
            string removido4 = teste.RemoverInicio();
            Console.WriteLine($"Site removido: {removido4}");
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }

        break;
    case 5:
        try
```

```
            {
                string removido5 = teste.RemoverFim();
                Console.WriteLine($"Site removido: {removido5}");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }

            break;
        case 6:
            Console.WriteLine("Informe a posicao:");
            int posicao6 = Convert.ToInt32(Console.ReadLine());
            try
            {
                string removido6 = teste.RemoverPosicao(posicao6);
                Console.WriteLine($"Site removido: {removido6}");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }

            break;
        case 7:
            teste.Mostrar();
            break;
        case 8:
            Console.WriteLine("Informe o nome do site:");
            string nomePesquisa = Console.ReadLine();
            string linkEncontrado = teste.PesquisarLink(nomePesquisa);
            Console.WriteLine($"Link: {linkEncontrado}");
            break;
        case 9:
            Console.Write("O programa sera encerrado.");
            continuar = false;
            break;
        default:
            continuar = false;
            break;
    }
  }
 }
}
```

Ex7:

```
using System;

namespace CSharpLista7
{
   public class CelulaDupla
   {
      public string Musica { get; set; }
      public CelulaDupla Anterior { get; set; }
      public CelulaDupla Proximo { get; set; }

      public CelulaDupla(string musica)
      {
         Musica = musica;
         Anterior = null;
         Proximo = null;
      }
   }

   public class ListaDupla
   {
      private CelulaDupla inicio;
      private CelulaDupla fim;

      public ListaDupla()
      {
         inicio = fim = null;
      }

      public void InserirInicio(string musica)
      {
         CelulaDupla nova = new CelulaDupla(musica);
         if (inicio == null)
         {
            inicio = fim = nova;
         }
         else
         {
            nova.Proximo = inicio;
            inicio.Anterior = nova;
            inicio = nova;
         }
      }

      public void InserirFim(string musica)
      {
         CelulaDupla nova = new CelulaDupla(musica);
         if (inicio == null)
```

```
    {
        inicio = fim = nova;
    }
    else
    {
        fim.Proximo = nova;
        nova.Anterior = fim;
        fim = nova;
    }
}

public void InserirPosicao(string musica, int posicao)
{
    if (posicao == 0)
    {
        InserirInicio(musica);
        return;
    }

    CelulaDupla atual = inicio;
    int i = 0;
    while (i < posicao - 1 && atual != null)
    {
        atual = atual.Proximo;
        i++;
    }

    if (atual == fim)
    {
        InserirFim(musica);
    }
    else if (atual != null)
    {
        CelulaDupla nova = new CelulaDupla(musica);
        nova.Proximo = atual.Proximo;
        atual.Proximo.Anterior = nova;
        nova.Anterior = atual;
        atual.Proximo = nova;
    }
}

public string RemoverInicio()
{
    if (inicio == null) throw new Exception("Lista vazia");
    string musica = inicio.Musica;
    inicio = inicio.Proximo;
    if (inicio != null)
    {
```

```
        inicio.Anterior = null;
    }
    else
    {
        fim = null;
    }

    return musica;
}

public string RemoverFim()
{
    if (fim == null) throw new Exception("Lista vazia");
    string musica = fim.Musica;
    fim = fim.Anterior;
    if (fim != null)
    {
        fim.Proximo = null;
    }
    else
    {
        inicio = null;
    }

    return musica;
}

public string RemoverPosicao(int posicao)
{
    if (inicio == null) throw new Exception("Lista vazia");
    if (posicao == 0) return RemoverInicio();

    CelulaDupla atual = inicio;
    int i = 0;
    while (i < posicao && atual != null)
    {
        atual = atual.Proximo;
        i++;
    }

    if (atual == null) throw new Exception("Posição inválida");

    if (atual == fim) return RemoverFim();

    atual.Anterior.Proximo = atual.Proximo;
    atual.Proximo.Anterior = atual.Anterior;
    return atual.Musica;
}
```

```
public bool Remover(string musica)
{
    CelulaDupla atual = inicio;
    while (atual != null)
    {
        if (atual.Musica.Equals(musica))
        {
            if (atual == inicio)
            {
                RemoverInicio();
            }
            else if (atual == fim)
            {
                RemoverFim();
            }
            else
            {
                atual.Anterior.Proximo = atual.Proximo;
                atual.Proximo.Anterior = atual.Anterior;
            }

            return true;
        }

        atual = atual.Proximo;
    }

    return false;
}

public void Mostrar()
{
    CelulaDupla atual = inicio;
    while (atual != null)
    {
        Console.WriteLine(atual.Musica);
        atual = atual.Proximo;
    }
}

public void MostrarInverso()
{
    CelulaDupla atual = fim;
    while (atual != null)
    {
        Console.WriteLine(atual.Musica);
        atual = atual.Anterior;
```

```
    }
}

public bool Pesquisar(string musica)
{
    CelulaDupla atual = inicio;
    while (atual != null)
    {
        if (atual.Musica.Equals(musica))
        {
            return true;
        }

        atual = atual.Proximo;
    }

    return false;
}

public string PesquisarAnterior(string musica)
{
    CelulaDupla atual = inicio;
    while (atual != null && atual.Proximo != null)
    {
        if (atual.Proximo.Musica.Equals(musica))
        {
            return atual.Musica;
        }

        atual = atual.Proximo;
    }

    return null;
}

public string PesquisarPosterior(string musica)
{
    CelulaDupla atual = inicio;
    while (atual != null && atual.Proximo != null)
    {
        if (atual.Musica.Equals(musica))
        {
            return atual.Proximo.Musica;
        }

        atual = atual.Proximo;
    }
```

```
        return null;
    }
}


public class GerenciadorMusica
{
    public static void Main()
    {
        ListaDupla lista = new ListaDupla();
        bool continuar = true;

        while (continuar)
        {
            Console.WriteLine("Menu:");
            Console.WriteLine("1)  Inserir uma musica no final da lista");
            Console.WriteLine("2)  Inserir uma musica no inicio da lista");
            Console.WriteLine("3)  Inserir uma musica numa posicao especifica da lista");
            Console.WriteLine("4)  Remover a musica do inicio da lista");
            Console.WriteLine("5)  Remover a musica do final da lista");
            Console.WriteLine("6)  Remover uma musica de uma posicao especifica da lista");
            Console.WriteLine("7)  Remover uma musica especifica");
            Console.WriteLine("8)  Listar todas as musicas da lista");
            Console.WriteLine("9)  Listar todas as musicas da lista na ordem inversa");
            Console.WriteLine("10) Pesquisar uma musica na lista");
            Console.WriteLine("11) Pesquisar musica anterior");
            Console.WriteLine("12) Pesquisar musica posterior");
            Console.WriteLine("13) Encerrar o programa");
            int opcao = int.Parse(Console.ReadLine());

            switch (opcao)
            {
                case 1:
                    Console.WriteLine("Informe a musica");
                    string musica1 = Console.ReadLine();
                    lista.InserirFim(musica1);
                    break;
                case 2:
                    Console.WriteLine("Informe a musica");
                    string musica2 = Console.ReadLine();
                    lista.InserirInicio(musica2);
                    break;
                case 3:
                    Console.WriteLine("Informe a musica");
                    string musica3 = Console.ReadLine();
                    Console.WriteLine("Informe a posicao");
                    int pos3 = int.Parse(Console.ReadLine());
                    lista.InserirPosicao(musica3, pos3);
```

```
    break;
case 4:
    Console.WriteLine("Musica removida: " + lista.RemoverInicio());
    break;
case 5:
    Console.WriteLine("Musica removida: " + lista.RemoverFim());
    break;
case 6:
    Console.WriteLine("Informe a posicao");
    int pos6 = int.Parse(Console.ReadLine());
    Console.WriteLine("Musica removida: " + lista.RemoverPosicao(pos6));
    break;
case 7:
    Console.WriteLine("Informe a musica");
    string musica7 = Console.ReadLine();
    if (lista.Remover(musica7))
    {
        Console.WriteLine("Musica removida");
    }
    else
    {
        Console.WriteLine("Musica nao encontrada");
    }

    break;
case 8:
    Console.WriteLine("Lista:");
    lista.Mostrar();
    break;
case 9:
    Console.WriteLine("Lista - ordem inversa:");
    lista.MostrarInverso();
    break;
case 10:
    Console.WriteLine("Informe a musica");
    string musica10 = Console.ReadLine();
    if (lista.Pesquisar(musica10))
    {
        Console.WriteLine("A musica esta na lista");
    }
    else
    {
        Console.WriteLine("A musica nao consta na lista");
    }

    break;
case 11:
    Console.WriteLine("Informe a musica");
```

```
                string musica11 = Console.ReadLine();
                string anterior = lista.PesquisarAnterior(musica11);
                if (anterior != null)
                {
                    Console.WriteLine("Musica anterior: " + anterior);
                }
                else
                {
                    Console.WriteLine("Nao ha musica anterior");
                }

                break;
            case 12:
                Console.WriteLine("Informe a musica");
                string musica12 = Console.ReadLine();
                string posterior = lista.PesquisarPosterior(musica12);
                if (posterior != null)
                {
                    Console.WriteLine("Musica anterior: " + posterior);
                }
                else
                {
                    Console.WriteLine("Nao ha musica posterior");
                }

                break;
            case 13:
                continuar = false;
                Console.Write("O programa sera encerrado.");
                break;
            default:
                Console.WriteLine("Opcao invalida.");
                break;
        }
    }
  }
 }
}
```