```
// LISTA ENCADEADA
class Program
{
    static void Main(string[] args)
    {
        var lista = new ListaEncadeada<int>();

        lista.AdicionarInicio(3);
        lista.AdicionarFinal(5);
        lista.AdicionarFinal(7);
        lista.AdicionarInicio(1);

        lista.RemoverInicio();
        lista.RemoverFinal();

        lista.ParaCada(x => Console.Write($"{x} "));
        Console.WriteLine();

        int soma = 0;
        lista.ParaCada(x => soma += x);
        Console.WriteLine($"Soma: {soma}");
    }
}
class No<T>
{
    private T valor;
    public T Valor
    {
        get{return valor;}
        set{valor = value;}
    }

    private No<T>? proximo = null;

    public No<T>? Proximo
    {
        get{return proximo;}
        set{proximo = value;}
    }
}

class ListaEncadeada<T>
{
    public void AdicionarInicio(T valor)
    {
        var novoNo = new No<T> {Valor = valor};

        if(primeiro == null)
        {
            primeiro = ultimo = novoNo;
        }
        else
        {
            novoNo.Proximo = primeiro;
            primeiro = novoNo;
        }
    }

    public void AdicionarFinal(T valor)
    {
        var novoNo = new No<T> {Valor = valor};

        if(ultimo == null)
        {
            ultimo = primeiro = novoNo;
        }
        else
        {
            ultimo.Proximo = novoNo;
            ultimo = novoNo;
        }
    }

    public void RemoverInicio()
    {
        if(primeiro == null)
        {
            throw new Exception("Nao é possivel remover no
inicio!");
        }

        primeiro = primeiro.Proximo;

        if(primeiro == null)
        {
            ultimo = null;
        }
    }

    public void RemoverFinal()
    {
        if(primeiro == null)
        {
            throw new Exception("Nao é possivel remover no
fim!");
        }

        if(primeiro.Proximo == null)
        {
            primeiro = null;
        }
        else
        {
            var penultimo = primeiro;

            while(penultimo?.Proximo?.Proximo != null)
            {
                penultimo = penultimo.Proximo;
            }

            if(penultimo != null)
            {
                penultimo.Proximo = null;
                ultimo = penultimo;
            }
        }
    }

    public void ParaCada(Action<T> visitar)
    {
        var atual = primeiro;

        while(atual != null)
        {
            visitar(atual.Valor);
            atual = atual.Proximo;
        }
    }

    private No<T>? primeiro = null;
    private No<T> ultimo = null;
}
```

```
// LISTA CONTIGUA
using System;

class Program
{
    static void Main(string[] args)
    {
        var lista = new ListaContigua();

        lista.AdicionarInicio(6);
        lista.AdicionarFinal(4);
        lista.AdicionarFinal(2);
        lista.AdicionarInicio(8);
        Imprimir(lista);

        lista.RemoverFinal();
        lista.RemoverInicio();

        Imprimir(lista);
    }

    static void Imprimir(ListaContigua lista)
    {
        for(int i = 0; i < lista.Tamanho; i++)
        {
            Console.Write("{0} ", lista.Get(i));
        }
        Console.WriteLine();
    }
}

class ListaContigua
{
    public int Tamanho {get => fim - inicio + 1;}

    private int fim;
    private int inicio;
    private double[] valores;


    public ListaContigua() : this(100) {}

    public ListaContigua(int tamanho)
    {
        inicio = tamanho / 2;
        fim = inicio - 1;
        valores = new double[tamanho];
    }

    public void AdicionarInicio(double valor)
    {
        if(inicio == 0)
        {
            throw new Exception("Nao é possivel
adicionar no incio!");
        }

        valores[--inicio] = valor;
    }

    public void AdicionarFinal(double valor)
    {
        if(fim == valores.Length - 1)
        {
            throw new Exception("Nao é possivel
adicionar no inicio!");
        }

        valores[++fim] = valor;
    }

    public void RemoverInicio()
    {
        if(Tamanho == 0)
        {
            throw new Exception("Nao existe
elemento a ser removido");
        }

        inicio++;
    }

    public void RemoverFinal()
    {
        if(Tamanho == 0)
        {
            throw new Exception("Nao existe
elemento a ser removido");
        }

        fim--;
    }

    public double Get(int indice)
    {
        if(indice < 0 || indice >= Tamanho)
        {
            throw new Exception("Indice
invalido!");
        }

        return valores[inicio + indice];
    }
}
```

```
// FILA
class Program
{
    static void Main(string[] args)
    {
        var fila = new Fila<int>();

        fila.Adicionar(3);
        fila.Adicionar(2);
        fila.Adicionar(1);

        while (fila.Inicio != null)
        {
            Console.WriteLine(fila.Inicio.Valor);
            fila.Remover();
        }
    }
}

class Fila<T>
{
    public void Adicionar(T valor)
    {
        var novoNo = new No<T>
        {
            Valor = valor
        };

        if(final == null)
        {
            final = inicio = novoNo;
        }
        else
        {
            final.Proximo = novoNo;
            final = novoNo;
        }
    }

    public void Remover()
    {
        if(inicio == null)
        {
            throw new Exception("Fila vazia!");
        }
        inicio = inicio.Proximo;

        if(inicio == null)
        {
            final = null;
        }
    }

    public No<T>? Inicio => inicio;

    private No<T> inicio = null;
    private No<T> final = null;
}
```

```
// PILHA
class Program
{
    static void Main(string[] args)
    {
        var pilha = new Pilha<int>();

        pilha.Empilhar(3);
        pilha.Empilhar(2);
        pilha.Empilhar(1);

        while(pilha.Topo != null)
        {
            Console.WriteLine(pilha.Topo.Valor);
            pilha.Desempilhar();
        }
    }
}

class Pilha<T>
{
    public void Empilhar(T valor)
    {
        var novoNo = new No<T> {Valor = valor};

        novoNo.Proximo = topo;
        topo = novoNo;
    }

    public void Desempilhar()
    {
        if(topo == null)
        {
            throw new Exception("Pilha vazia!");
        }

        topo = topo.Proximo;
    }

    public No<T>? Topo => topo;

    private No<T>? topo = null;
}


// PILHA E FILA USAM A MESMA CLASSE NO DA LISTA ENCADEADA
```