İKİ YÖNLÜ LİNKED LİST YAPISI

```
static void Main(string[] args)
{
    BagliListe liste = new BagliListe();

    // Örnek işlemler
    liste.BasaEkle(10);
    liste.SonaEkle(20);
    liste.SonaEkle(30);
    liste.BasaEkle(5);
    liste.DegerdenSonraEkleme(20, 25);
    liste.DegerdenOnceEkleme(10, 8);

    Console.WriteLine("Listeleme:");
    liste.Listele();

    Console.WriteLine("\nArama (25):");
    int idx = liste.Arama(25);
    Console.WriteLine(idx >= 0 ? $"Bulundu. Konum: {idx}" : "Bulunamadı.");

    Console.WriteLine("\nBaştan silme:");
    liste.BastanSilme();
    liste.Listele();

    Console.WriteLine("\nSondan silme:");
    liste.SondanSilme();
    liste.Listele();

    Console.WriteLine("\nAradan silme (20):");
    liste.AradanSilme(20);
    liste.Listele();

    Console.WriteLine("\nDizide atma:");
    int[] dizi = liste.ToArray();
    Console.WriteLine(string.Join(", ", dizi));

    Console.WriteLine("\nTümünü silme:");
    liste.TumunuSil();
    liste.Listele();

    Console.WriteLine("\nİşlem tamam. Çıkmak için bir tuşa basın...");
    Console.ReadKey();
}
//düğüm oluşturduk
public class Dügüm
{
    public int veri;
    public Dügüm Sonraki; //bunlar ipleri gösterir sonraki ip veya önceki ip
    public Dügüm Onceki;
    public Dügüm(int veri)
    {
        this.veri = veri;
        this.Sonraki = null;
        this.Onceki = null;
    }
}
public class BagliListe
{
    public Dügüm bas;
```

```
    public Dügüm son;

    public BagliListe()
    {
        this.bas = null;
        this.son = null;
    }

    //basa ekle
    public void BasaEkle(int veri)
    {
        Dügüm yeniDügüm = new Dügüm(veri);
        if (bas == null)
        {
            bas = yeniDügüm;
            son = yeniDügüm;
        }
        else
        {
            yeniDügüm.Sonraki = bas;
            bas.Onceki = yeniDügüm;
            bas = yeniDügüm;
        }
        Console.WriteLine($"{veri} listeye başa eklendi.");
    }
    //sona ekleme
    public void SonaEkle(int veri)
    {
        Dügüm yeniDügüm = new Dügüm(veri);
        if (bas == null)
        {
            bas = yeniDügüm;
            son = yeniDügüm;
        }
        else
        {
            son.Sonraki = yeniDügüm;
            yeniDügüm.Onceki = son;
            son = yeniDügüm;
        }
        Console.WriteLine($"{veri} listeye sona eklendi.");
    }
    //değerden sonra ekleme
    public void DegerdenSonraEkleme(int hedefVeri, int eklenecekVeri)
    {
        Dügüm yeniDügüm = new Dügüm(eklenecekVeri);

        if (bas == null)
        {
            Console.WriteLine("Veri yok bu yüzden başa eklendi.");
            bas = yeniDügüm;
            son = yeniDügüm;
            return;
        }
        Dügüm mevcut = bas;
        while (mevcut != null && mevcut.veri != hedefVeri) //hedef düğümü arıyor
baştan başlatarak.
        {
            mevcut = mevcut.Sonraki;
        }

        if (mevcut == null) //hedef düğüm bulundu mu?
        {
```

```
            Console.WriteLine($"{hedefVeri} verisine sahip düğüm bulunamadı.");
            return;
        }
        if (mevcut == son) //hedef düğüm sonda mı kontrol ediyor.
        {
            SonaEkle(eklenecekVeri);
            return;
        }
        yeniDügüm.Sonraki = mevcut.Sonraki;
        yeniDügüm.Onceki = mevcut;
        mevcut.Sonraki.Onceki = yeniDügüm;
        mevcut.Sonraki = yeniDügüm;
        Console.WriteLine($"{eklenecekVeri} listeye {hedefVeri} verisinden sonra
eklendi.");
    }
    //değerden önce ekleme
    public void DegerdenOnceEkleme(int hedefVeri, int eklenecekVeri)
    {
        Dügüm yeniDügüm = new Dügüm(eklenecekVeri);
        if (bas == null)
        {
            Console.WriteLine("Veri yok bu yüzden başa eklendi.");
            bas = yeniDügüm;
            son = yeniDügüm;
            return;
        }

        Dügüm mevcut = bas;
        while (mevcut != null && mevcut.veri != hedefVeri)
        {
            mevcut = mevcut.Sonraki;
        }

        if (mevcut == null)
        {
            Console.WriteLine($"{hedefVeri} verisine sahip düğüm bulunamadı
{eklenecekVeri} eklenemedi");
            return;
        }

        if (mevcut == bas)
        {
            yeniDügüm.Sonraki = bas;
            bas.Onceki = yeniDügüm;
            bas = yeniDügüm;
            Console.WriteLine($"{eklenecekVeri} listeye {hedefVeri} verisinden
önce eklendi.");
            return;
        }

        yeniDügüm.Sonraki = mevcut;
        yeniDügüm.Onceki = mevcut.Onceki;
        mevcut.Onceki.Sonraki = yeniDügüm;
        mevcut.Onceki = yeniDügüm;
        Console.WriteLine($"{eklenecekVeri} listeye {hedefVeri} verisinden önce
eklendi.");
    }

    // Baştan silme
    public void BastanSilme()
    {
        if (bas == null)
        {
```

```
            Console.WriteLine("Liste boş silme işlemi yapılamaz.");
            return;
        }
        if (bas == son) // sadece bir eleman varsa diye kontrol ediyor.
        {
            Console.WriteLine($"{bas.veri} baştan silindi, liste boş kaldı.");
            bas = null;
            son = null;
            return;
        }
        Console.WriteLine($"{bas.veri} baştan silindi.");
        bas = bas.Sonraki;
        if (bas != null)
        {
            bas.Onceki = null;
        }
    }

    // Sondan silme
    public void SondanSilme()
    {
        if (bas == null)
        {
            Console.WriteLine("Liste boş silme işlemi yapılamaz.");
            return;
        }
        if (bas == son) //tek eleman mı var acaba diye bir kontrol yapılıyor.
        {
            Console.WriteLine($"{son.veri} sondan silindi, liste boş kaldı.");
            bas = null;
            son = null;
            return;
        }
        Console.WriteLine($"{son.veri} sondan silindi.");
        son = son.Onceki;
        if (son != null)
        {
            son.Sonraki = null;
        }
    }

    // Aradan arayarak silme (ilk bulunanı siler)
    //bu metod listede soldan sağa sırayla arama yapar,
    //verilen değere sahip ilk düğümü bulursa onu siler
    public void AradanSilme(int veri)
    {
        if (bas == null)
        {
            Console.WriteLine("Liste boş, silme yapılamaz.");
            return;
        }
        Dügüm mevcut = bas;
        while (mevcut != null && mevcut.veri != veri)
        {
            mevcut = mevcut.Sonraki;
        }
        if (mevcut == null)
        {
            Console.WriteLine($"{veri} bulunamadı, silme yapılmadı.");
            return;
        }
        if (mevcut == bas)
        {
```

```
            BastanSilme();
            return;
        }
        if (mevcut == son)
        {
            SondanSilme();
            return;
        }
        // Ara düğüm
        mevcut.Onceki.Sonraki = mevcut.Sonraki;
        mevcut.Sonraki.Onceki = mevcut.Onceki;
        Console.WriteLine($"{veri} aradan silindi.");
    }

    // Arama (ilk bulunanın indeksini döner, bulunamazsa -1)
    public int Arama(int veri)
    {
        Dügüm mevcut = bas;
        int idx = 0;
        while (mevcut != null)
        {
            if (mevcut.veri == veri)
            {
                return idx;
            }
            mevcut = mevcut.Sonraki;
            idx++;
        }
        return -1;
    }

    // Listeleme
    public void Listele()
    {
        if (bas == null)
        {
            Console.WriteLine("Liste boş.");
            return;
        }
        Dügüm mevcut = bas;
        List<int> elemanlar = new List<int>();
        while (mevcut != null)
        {
            elemanlar.Add(mevcut.veri);
            mevcut = mevcut.Sonraki;
        }
        Console.WriteLine(string.Join(" <-> ", elemanlar));
        //string.Join çok güzel bir şeye yarıyor. Elemanların arasına kolay bir
şekilde <-> bunu koymamıza yarıyor.
    }

    // Tümünü silme
    public void TumunuSil()
    {
        bas = null;
        son = null;
        Console.WriteLine("Liste tamamen silindi.");
    }

    // Tüm linked listi bir diziye atma
    public int[] ToArray()
    {
        List<int> temp = new List<int>();
```

```
            Dügüm mevcut = bas;
            while (mevcut != null)
            {
                temp.Add(mevcut.veri);
                mevcut = mevcut.Sonraki;
            }
            return temp.ToArray();
        }
    }
```