Stack Veri Yapısı Ödev

1. SORU

Öncelikle sabit dizi ile tanımlı bir çıkının içerdiği elamanları tanımlamak gerekir

```
public class Ornek {
    int içerik;
    public Ornek (int icerik){
            this.icerik=içerik;
    }
}
```

Sonraki adım tam sayılar içeren bir çıkının sabit dizi ile Tanımlanması olacaktır.

```
public class Cikin{
    Ornek dizi[];
    İnt ust;
    İnt N;
    public Cikin (int N ){
            dizi =new Ornek[N];
            this.N=N;
            ust=-1;
            }
    Ornek ust (){
            return dizi[ust];
                }
    boolean cikinDolu(){
            İf(ust==N-1)
```

```
                return true;
            else
                return false;
                    }
    boolean cikinBos(){
      if(ust==-1)
                return true;
       else
                return false;
                        }
}
```

Gerekli tanımlamaları yaptıktan sonra çıkına eleman ekleme ve silme işlemlerimizi gerçekleştirebiliriz

Çıkına Eleman Ekleme

Çıkına yeni bir eleman eklerken yapmamız gereken ust değişkenini değiştirmek ve yeni eklenen elemanı göstermesini sağlamaktır.

```
    void cikinEkle(Ornek yeni){
      if(!cikinDolu()){
                ust++;
                dizi[ust]=yeni;
                        }
      }
```

Çıkından Eleman Silme

Çıkından bir eleman silerken dikkat etmemiz gereken şey çıkının boş olup olmamasıdır.Eğer çıkın boş değilse ,çıkının son elemanı geri döndürülür ve çıkının üst değişkeni bir azaltılır.Çıkın boşsa fonksiyon NULL döndürür.

```
Ornek cikinSil(){
        İf(!cikinBos()){
                ust--;
                return dizi[ust+1];
                        }
        return null;
}
```

Çıkın İşlemleri Zaman Karmaşıklığı (Sabit Dizi)

Ekleme : O(1)

Silme : O(1)

2. SORU

Çıkın İşlemleri Zaman Karmaşıklığı (Bağlı Liste)

Ekleme : O(1)

Silme : O(1)

3. SORU

Alan karmaşıklığı

Bağlı liste :O(n)

Sabit Dizi:O(1)

Dizilerde uygulama başlatılmadan önce boyutunun belirlenmesi gerekir.

Bağlı listelerde ise böyle bir kısıtlama yoktur.

Burdan kullanılıcak verinin bilindigi durumlarda bağlı listeleri kullanmanın

daha mantıklı olacağını söyleyebilirim.

Bir diger yandan, Bağlantılı listenin bellek yükü genellikle her öğede fazladan bir işaretçinin depolanması nedeniyle toplam O (n) fazladan bellektir.

dinamik bir dizinin bellek kullanımı oldukça iyidir - dizi tamamen dolu olduğunda, örneğin, sadece O (1) fazladan ek yük vardır.

Dinamik belleginde alan maliyeti açısından böyle bir avantajı vardır.

Verilen probleme göre kullanacağımız yöntem değişir.