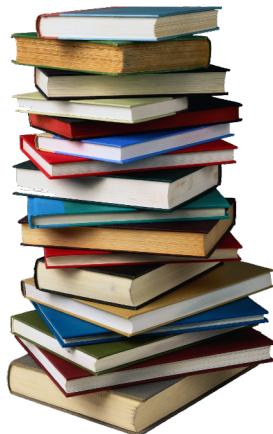


# **Yığın & Kuyruk**

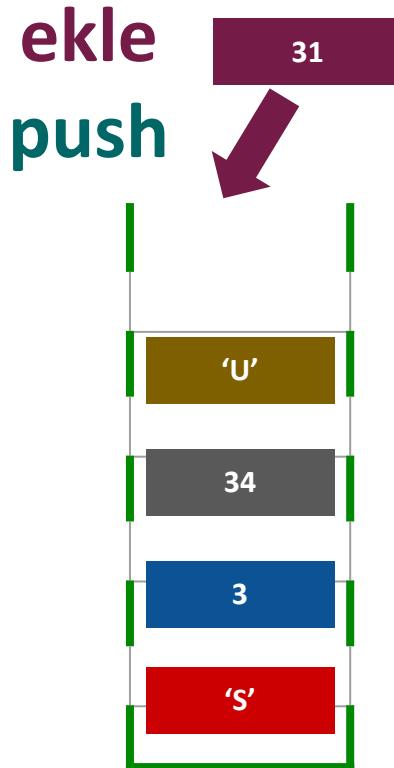


**Suhap SAHIN  
Onur GÖK**

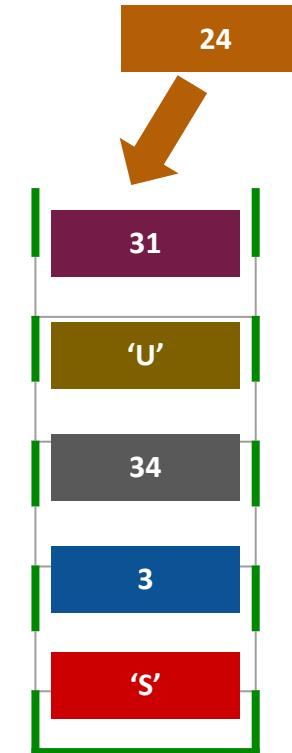
# Yığın (Stack)



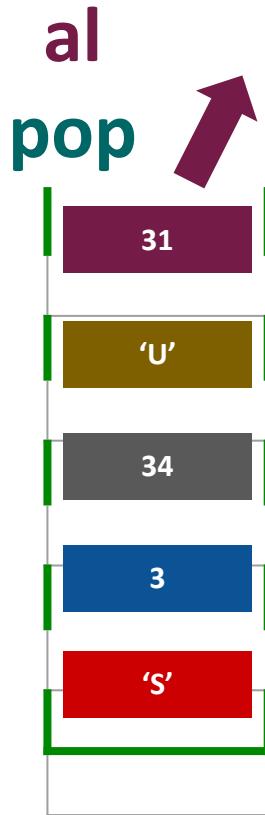
# Yığın işlemleri(SO)



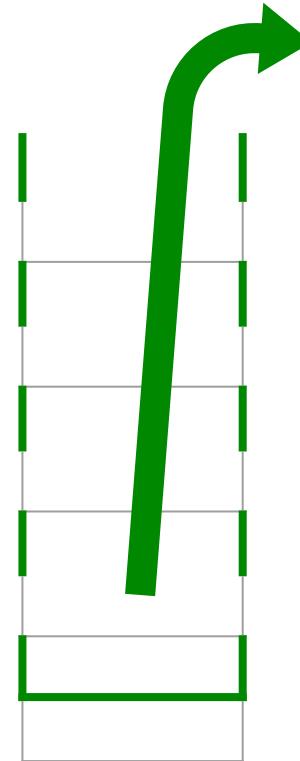
stack  
overflow



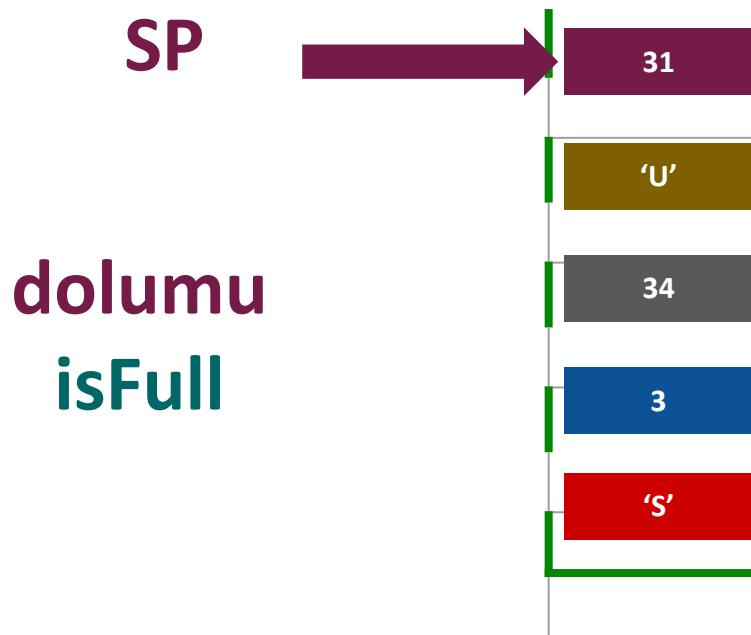
# Yığın işlemleri(SO)



stack  
underflow



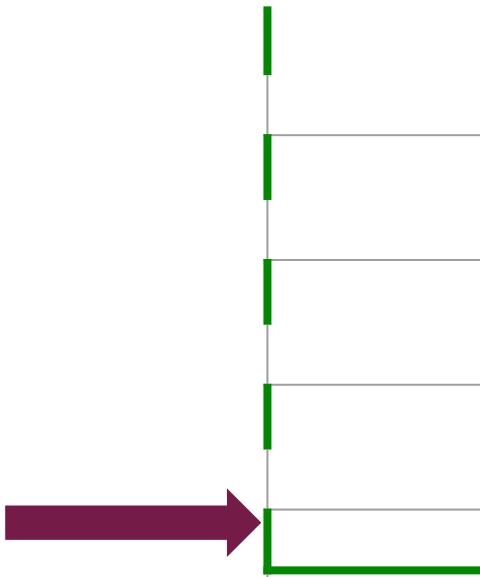
# Yığın işlemleri(SO)



# Yığın işlemleri(SO)

bosmu  
isEmpty

SP



# Yığın işlemleri(SO)

**ekle(), push()**

**O(1)**

**al(), pop()**

**bosmu(), isEmpty()**

**dolumu(), isFull()**



# Yığın Uygulamaları

Sembollerin esitlenmesi

```
#include <stdio.h>
int main () {
    for(int i=0;i<10;i++){
        // Kod
    }
}
```



# Yığın Uygulamaları

## Son-ek(Posfix)/Ön-ek(Prefix) ifadeleri

a op1 b op2 c op3 d

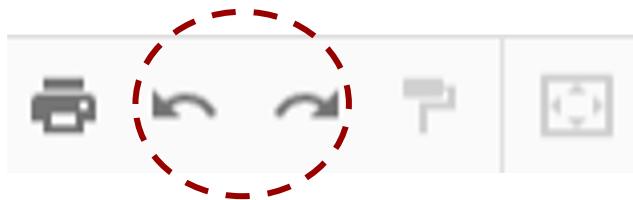
op1 = +

op2 = \*

op3 = +

# Yığın Uygulamaları

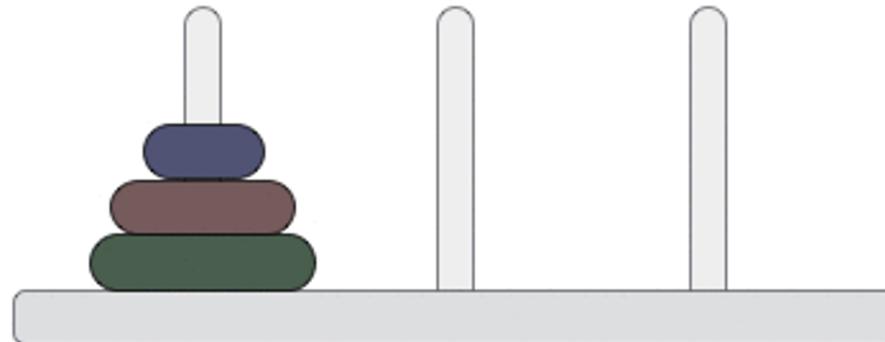
Bir çok programdaki ileri-al, geri-al (redo-undo) ve ileri-git, geri-git (forward-backward) özellikleri



# Yığın Uygulamaları

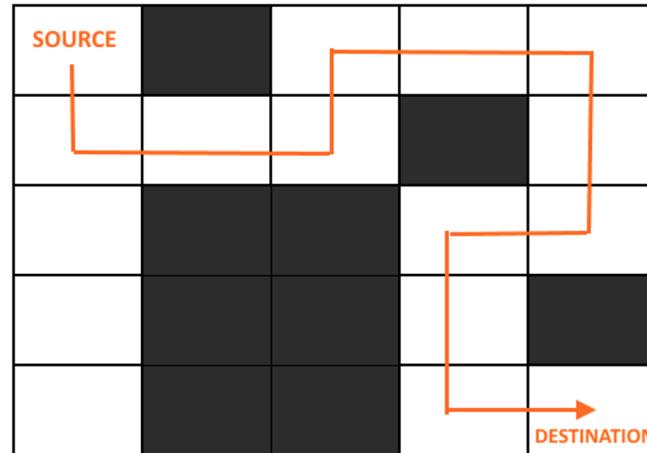
Tower of Hanoi, tree traversals ...

Step: 0

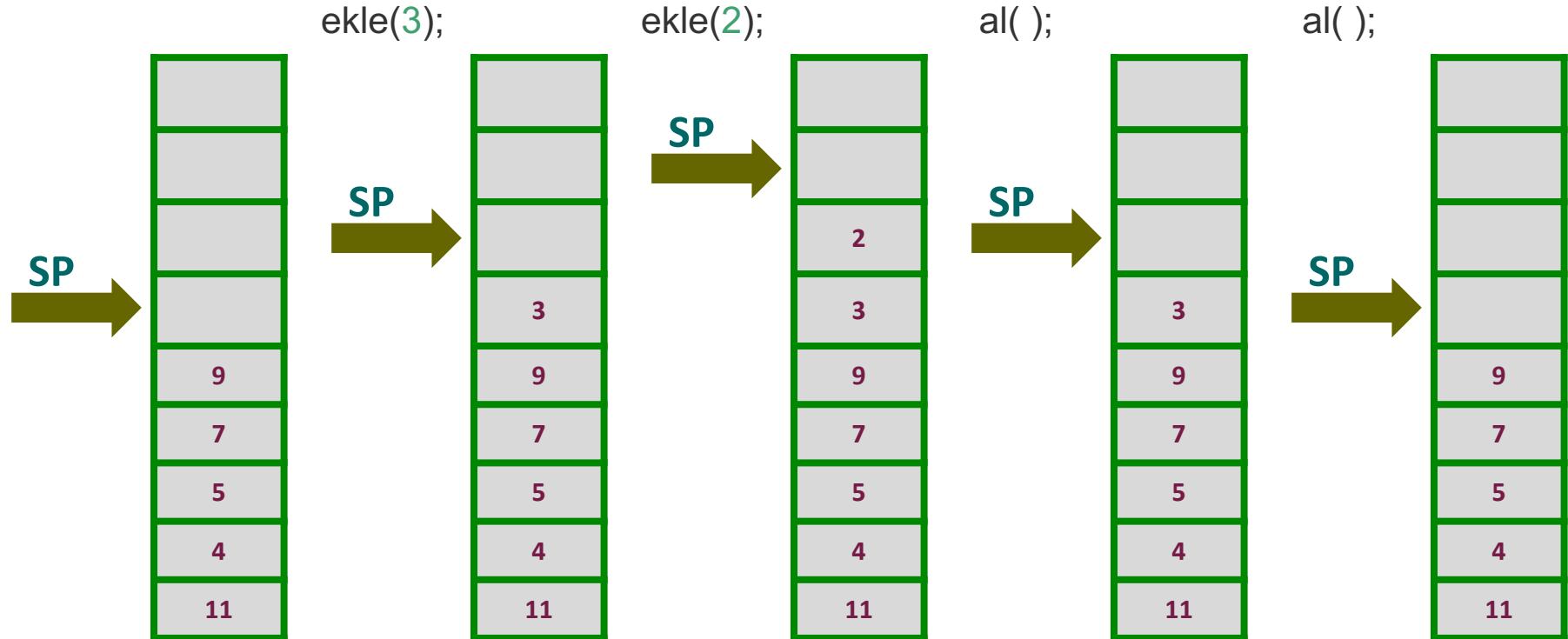


# Yığın Uygulamaları

# Backtracking, Knight tour problem, rat in a maze, ... sudoku solver

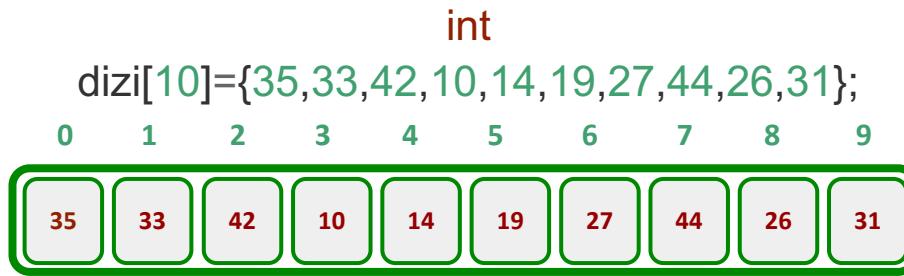


# Yığın işaretcisi (SP)

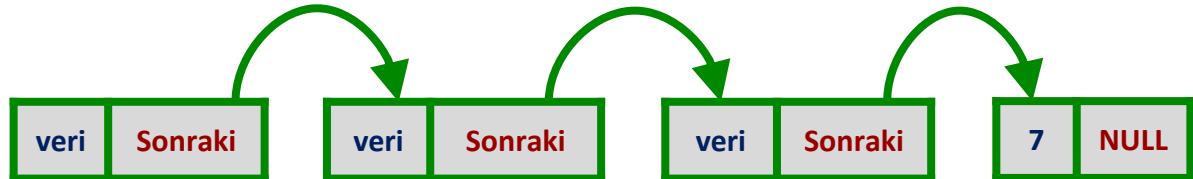


# Yığın Gerçekleştirmi

Dizi

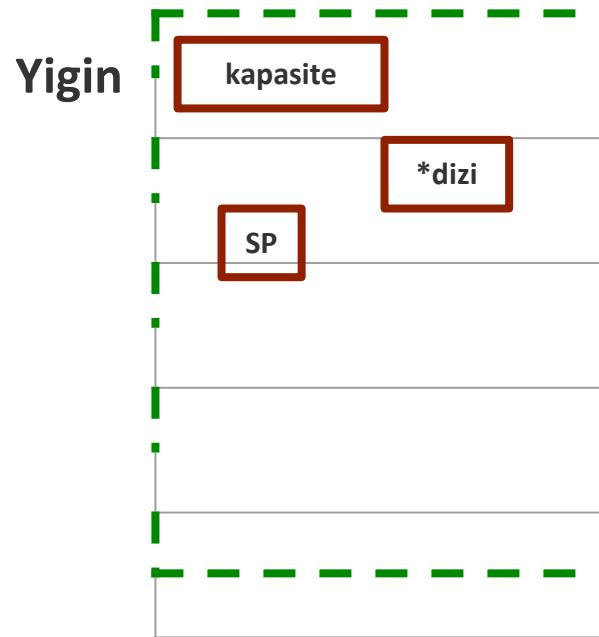


Baglantılı  
Listeler



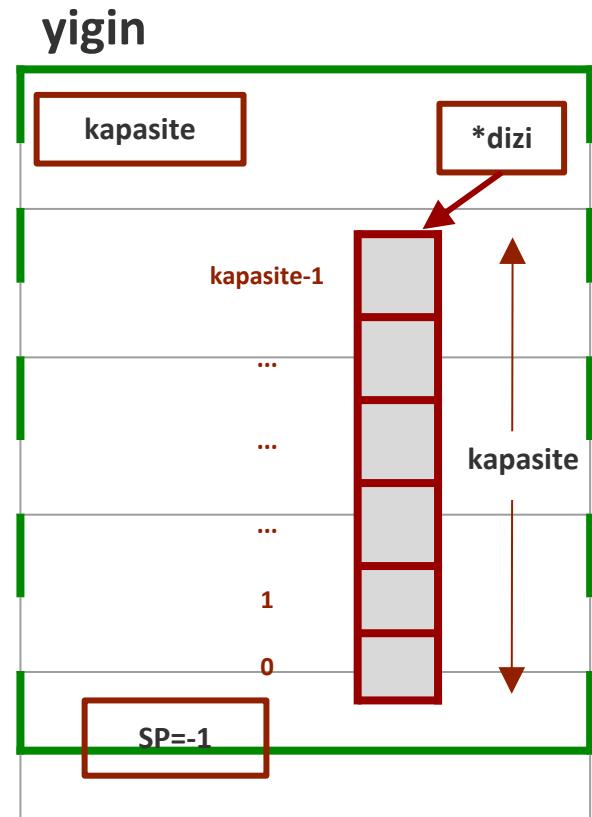
# Yığın(Dizi) & C

```
struct Yigin
{
    int SP;
    unsigned kapasite;
    int* dizi;
};
```



# Yığın(Dizi) & C

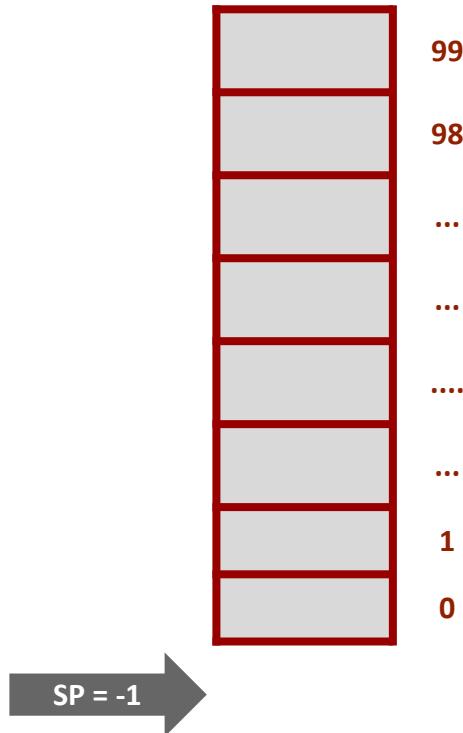
```
struct Yigin* yiginOlustur(unsigned kapasite)
{
    struct Yigin* yigin = (struct Yigin*) malloc(sizeof(struct Yigin));
    yigin->kapasite = kapasite;
    yigin->SP = -1;
    yigin->dizi = (int*) malloc(yigin->kapasite * sizeof(int));
    return yigin;
}
```



# Yığın(Dizi) & C

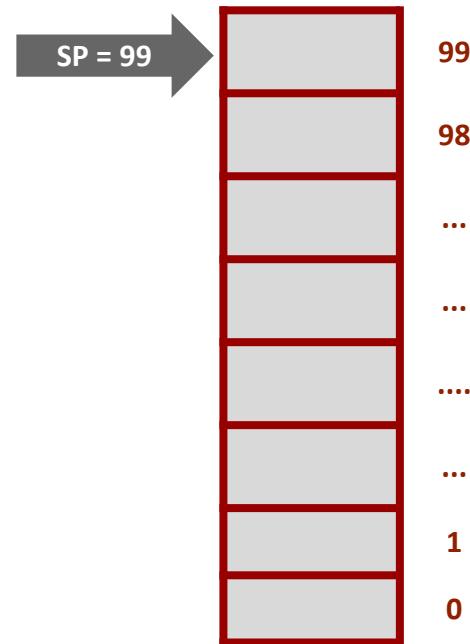
```
struct Yigin* yiginOlustur(unsigned kapasite)
{
    struct Yigin* yigin = (struct Yigin*) malloc(sizeof(struct Yigin));
    yigin->kapasite = kapasite;
    yigin->SP = -1;
    yigin->dizi = (int*) malloc(yigin->kapasite * sizeof(int));
    return yigin;
}
```

```
struct Yigin* yigin = yiginOlustur(100);
```



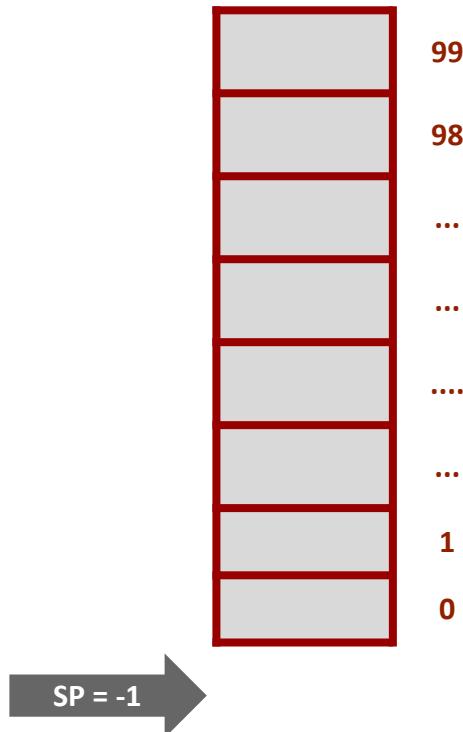
# Yığın(Dizi) & C

```
int dolumu(struct Yigin* yigin)
{ return yigin->SP == yigin->kapasite - 1; }
```



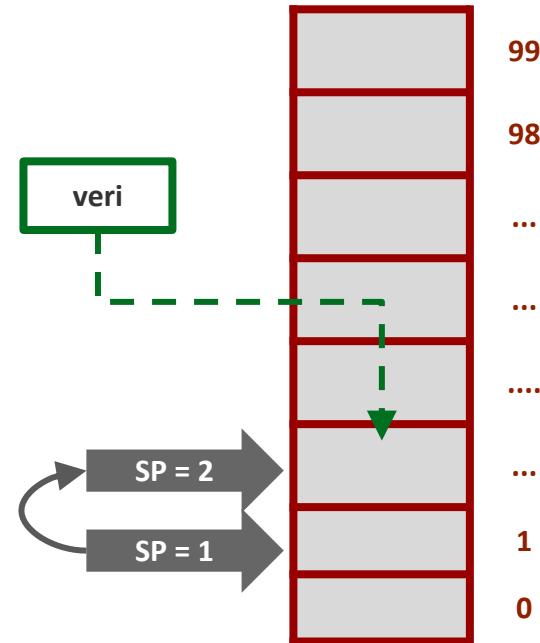
# Yığın(Dizi) & C

```
int bosmu(struct Yigin* yigin)
{ return yigin->SP == -1; }
```



# Yığın(Dizi) & C

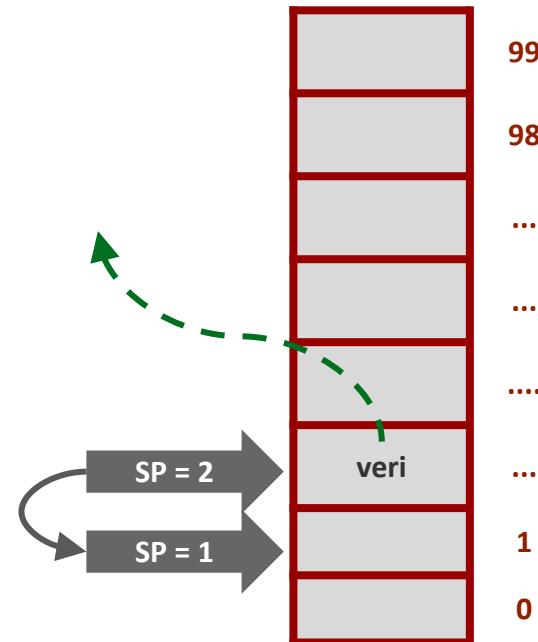
```
void ekle(struct Yigin* yigin, int veri)
{
    if (dolumu(yigin))
        return;
    yigin->dizi[++yigin->SP] = veri;
    printf("%d Yigina eklendi\n", veri);
}
ekle(yigin, 10);
```



# Yığın(Dizi) & C

```
int al(struct Yigin* yigin)
{
    if (bosmu(yigin))
        return INT_MIN;
    return yigin->dizi[yigin->SP--];
}

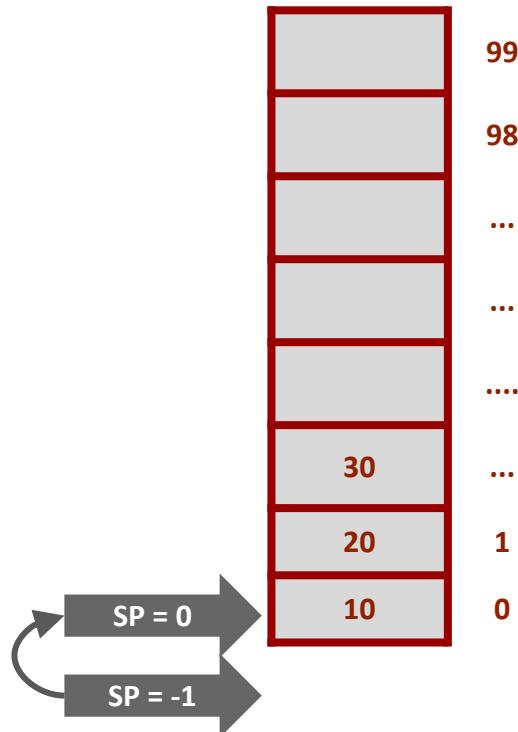
al(yigin);
```



# Yığın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);

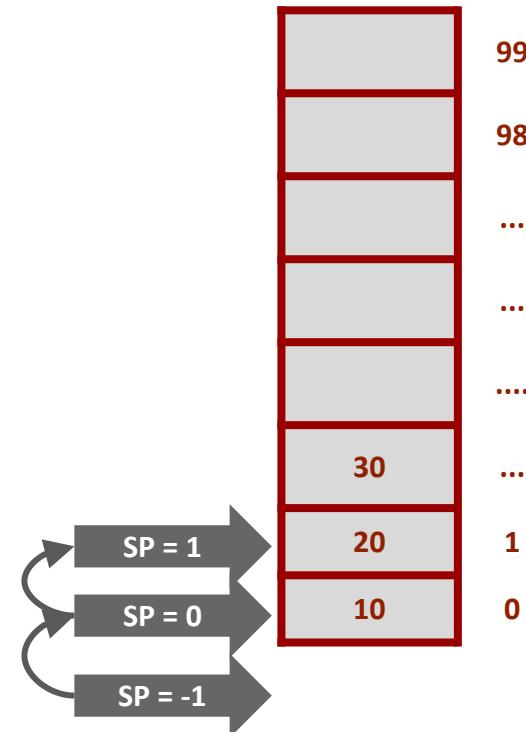
    return 0;
}
```



# Yığın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);

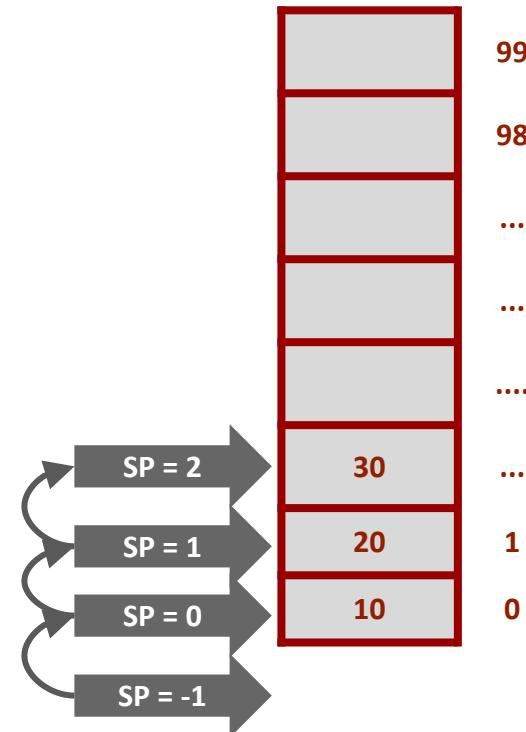
    return 0;
}
```



# Yığın(Dizi) & C

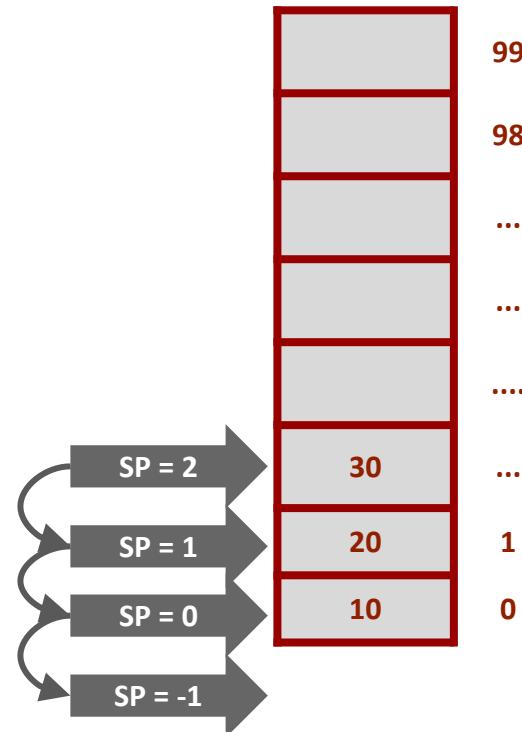
```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);
    ekle(yigin, 30);

    return 0;
}
```



# Yığın(Dizi) & C

```
int main()
{
    struct Yigin* yigin = yiginOlustur(100);
    ekle(yigin, 10);
    ekle(yigin, 20);
    ekle(yigin, 30);
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    printf("Yiginda %d alindi\n", al(yigin));
    return 0;
}
```



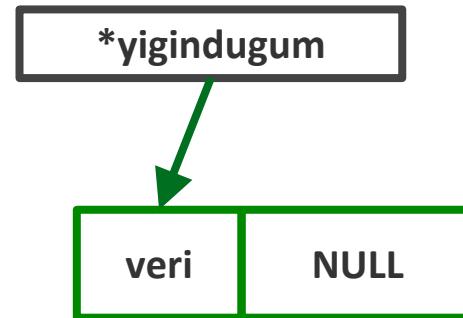
# Yığın(Bağlı Liste) & C

```
struct YiginDugum  
{  
    int veri;  
    struct YiginDugum* sonraki;  
};
```



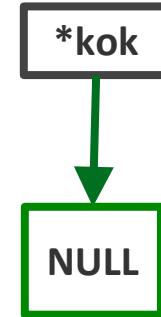
# Yığın(Bağlı Liste) & C

```
struct YiginDugum* YeniDugum(int veri)
{
    struct YiginDugum* yigindugum = (struct YiginDugum*) malloc(sizeof(struct YiginDugum));
    yigindugum->veri = veri;
    yigindugum->sonraki = NULL;
    return yigindugum;
}
```



# Yığın(Bağlı Liste) & C

```
struct YiginDugum* kok = NULL;
```



# Yığın(Bağlı Liste) & C

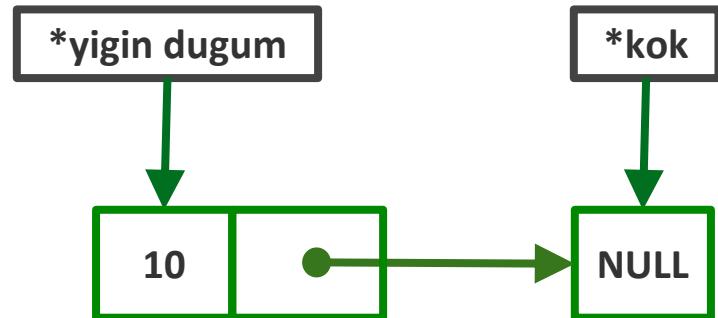
```
void ekle(struct YiginDugum** kok, int veri)
```

```
{
```

```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
    ekle(&kok, 10);
```



# Yığın(Bağlı Liste) & C

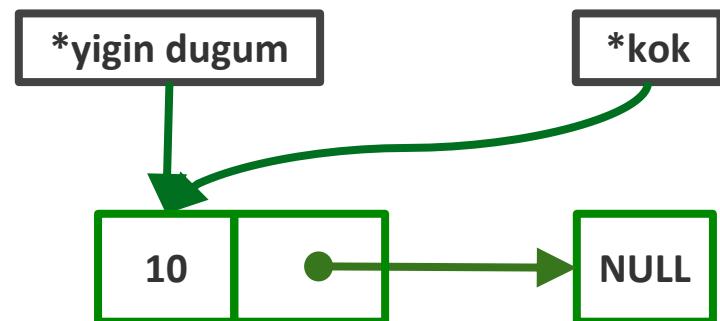
```
void ekle(struct YiginDugum** kok, int veri)
```

```
{
```

```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
ekle(&kok, 10);
```



# Yığın(Bağlı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
```

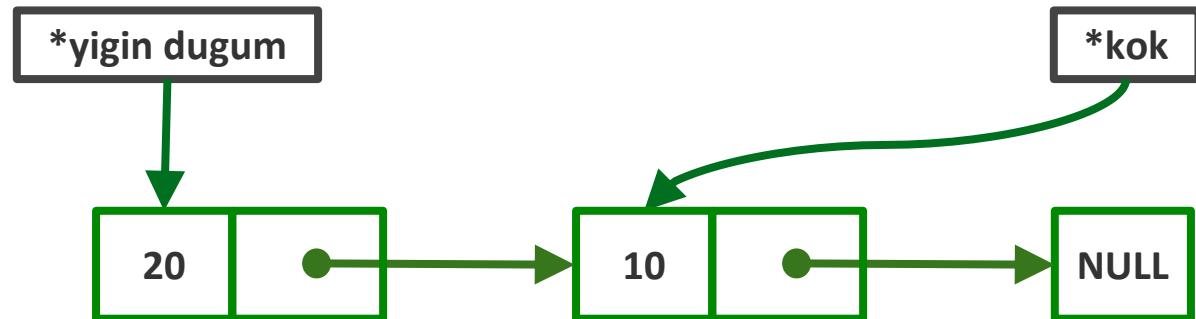
```
{
```

```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
ekle(&kok, 10);
```

```
ekle(&kok, 20);
```



# Yığın(Bağlı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
```

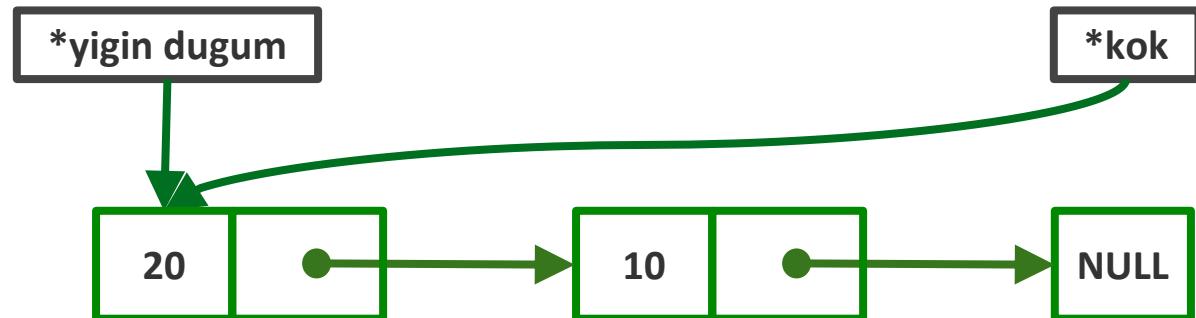
```
{
```

```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
ekle(&kok, 10);
```

```
ekle(&kok, 20);
```



# Yığın(Bağlı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
```

```
{
```

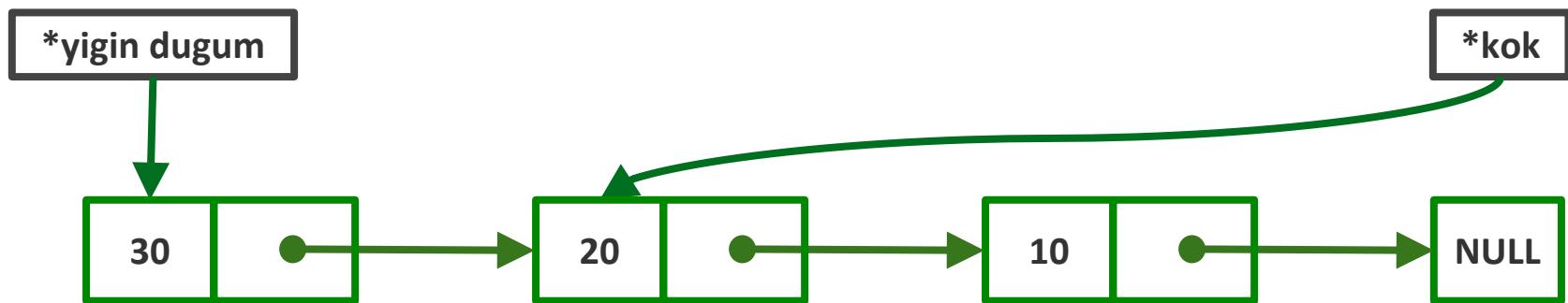
```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
ekle(&kok, 10);
```

```
ekle(&kok, 20);
```

```
ekle(&kok, 30);
```



# Yığın(Bağlı Liste) & C

```
void ekle(struct YiginDugum** kok, int veri)
```

```
{
```

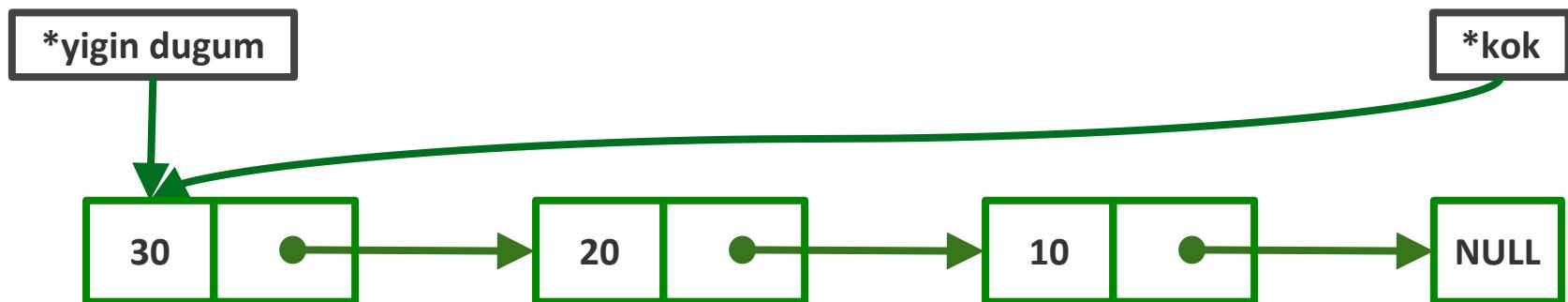
```
    struct YiginDugum* yigindugum = YeniDugum(veri);  
    yigindugum->sonraki = *kok;  
    *kok = yigindugum;  
    printf("%d verisine sahip dugum eklendi\n", veri);
```

```
}
```

```
ekle(&kok, 10);
```

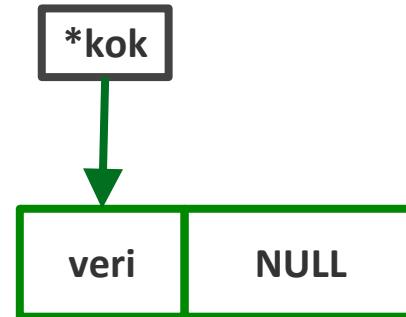
```
ekle(&kok, 20);
```

```
ekle(&kok, 30);
```



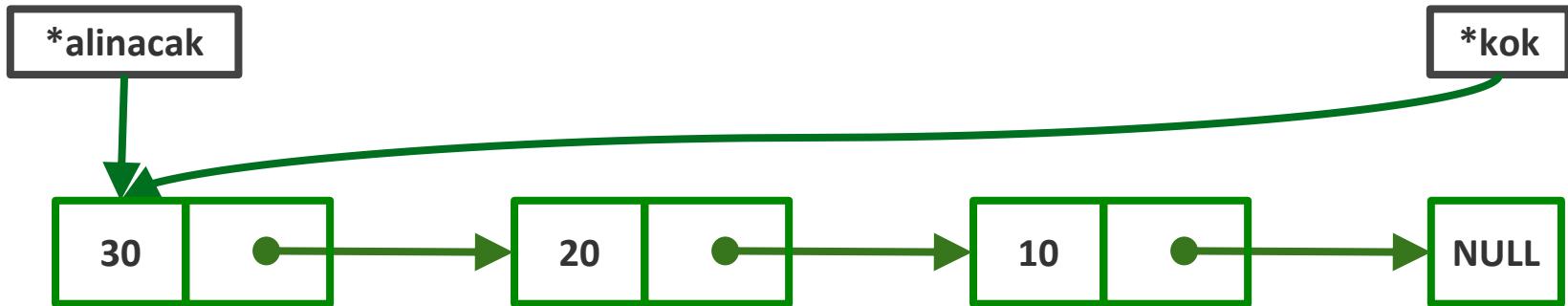
# Yığın(Bağlı Liste) & C

```
int bosmu(struct YiginDugum *kok)
{
    return !kok;
}
```



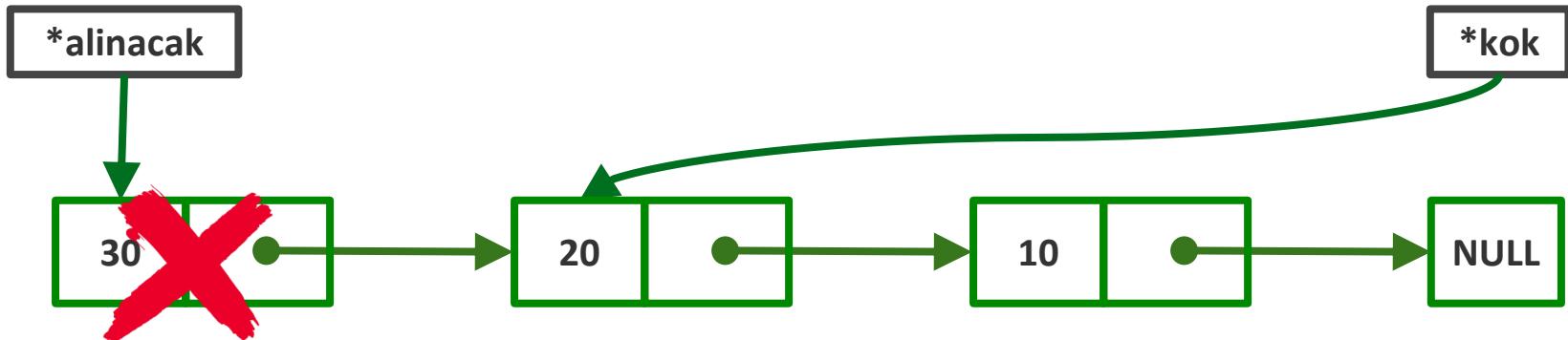
# Yığın(Bağlı Liste) & C

```
int al(struct YiginDugum** kok)
{
    if (bosmu(*kok))
        return INT_MIN;
    struct YiginDugum* alinacak = *kok;
    *kok = (*kok)->sonraki;
    int alinan = alinacak->veri;
    free(alinacak);
    return alinan;
}
```



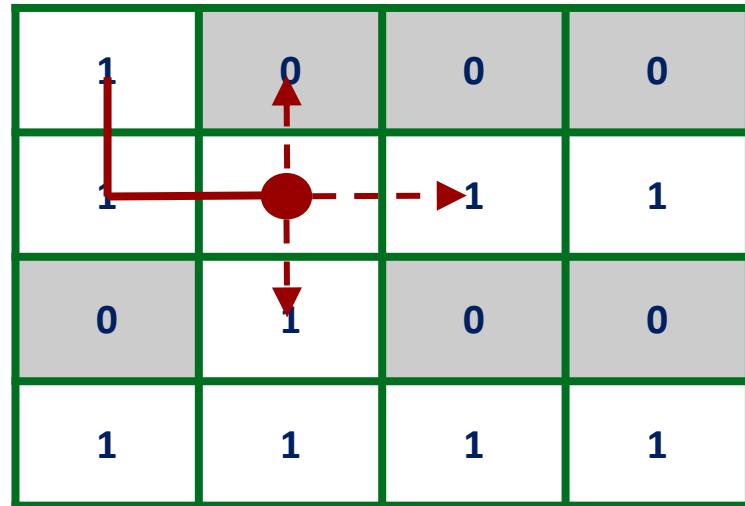
# Yığın(Bağlı Liste) & C

```
int al(struct YiginDugum** kok)
{
    if (bosmu(*kok))
        return INT_MIN;
    struct YiginDugum* alinacak = *kok;
    *kok = (*kok)->sonraki;
    int alinan = alinacak->veri;
    free(alinacak);
    return alinan;
}
```

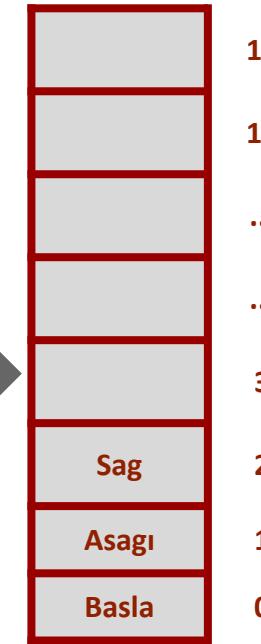


# Labirent

Fare

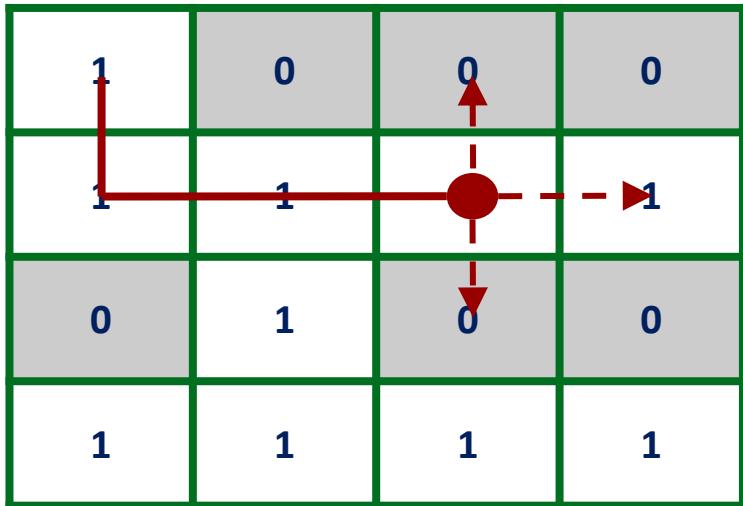


SP = 3 →

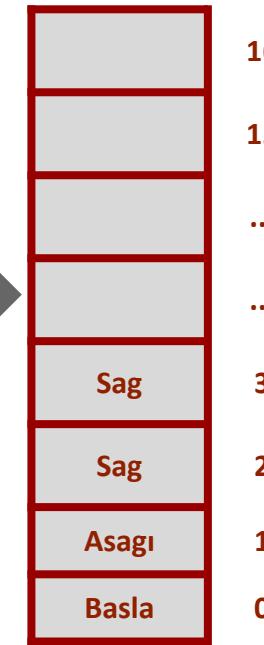


# Labirent

Fare

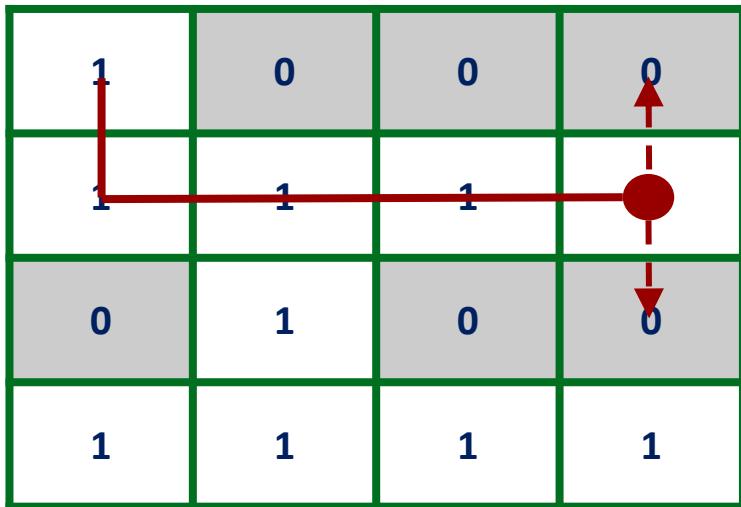


SP = 3



# Labirent

Fare



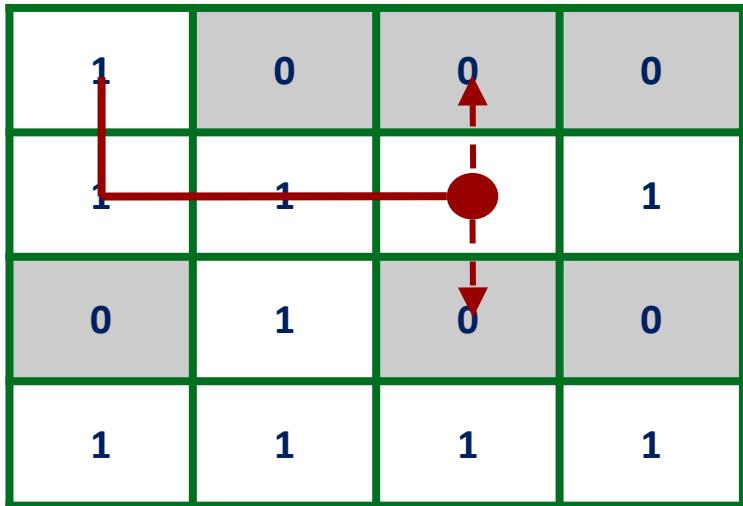
Peynir

SP = 3 →

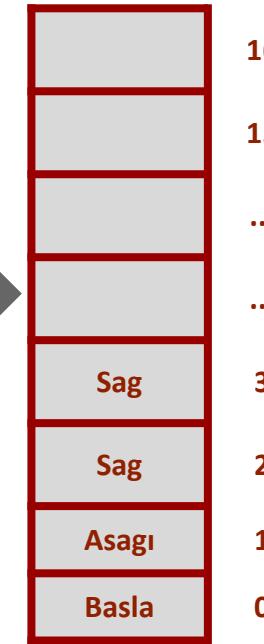


# Labirent

Fare

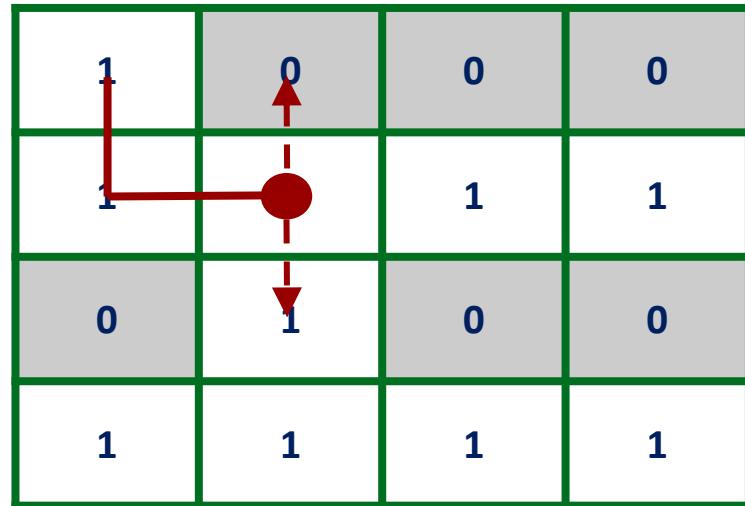


SP = 3 →

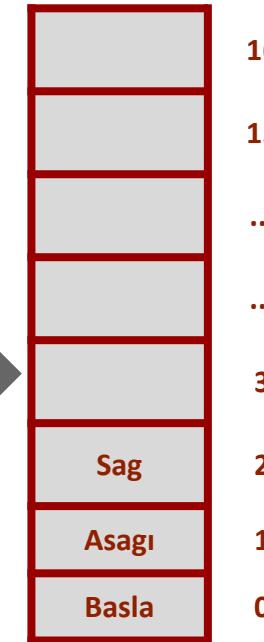


# Labirent

Fare

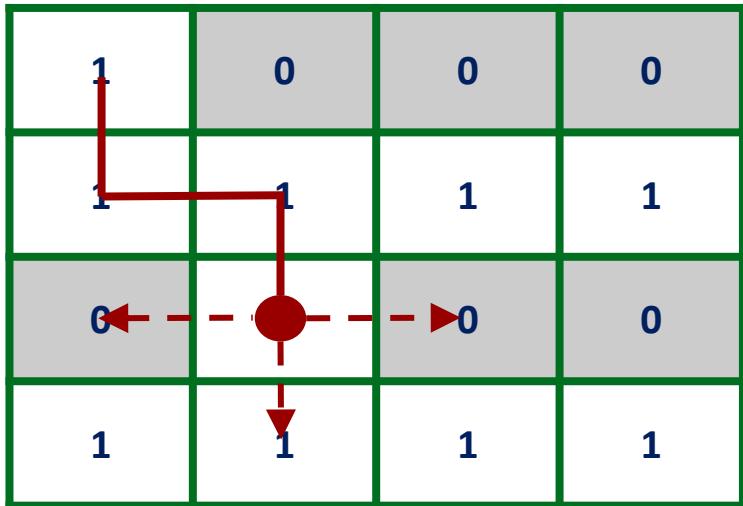


SP = 3 →



# Labirent

Fare

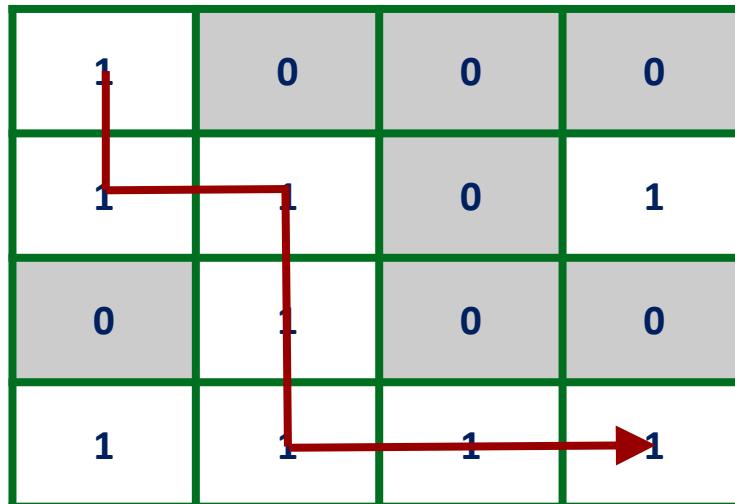


SP = 3 →



# Labirent

Fare



Peynir

# Kuyruk

Her iki ucu açık

Bir uçtan veri (**enqueue**) ekleme

Diger uçtan veri (**dequeue**) çıkarma

**First-In-First-Out**



# Gösterim



# Kuyruk Operasyonları

**enqueue(): Sona elemanı ekler**

**dequeue(): En öndeği elemanı çıkartır**

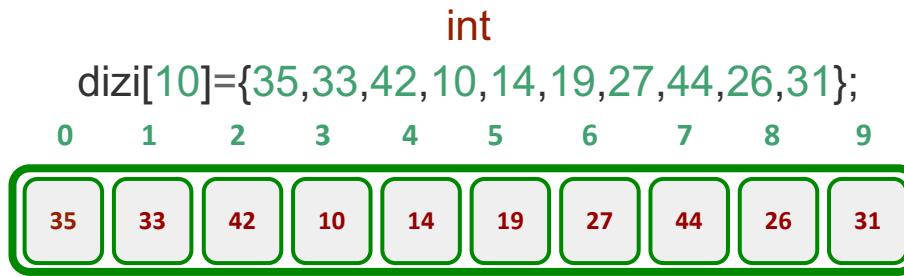
**isfull(): Kuyruk dolu mu?**

**isempty(): Kuyruk bos mu?**

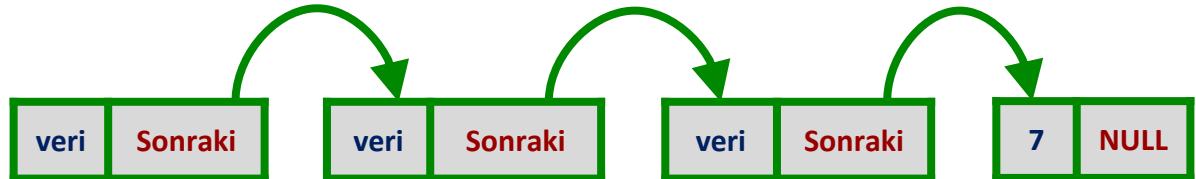


# Kuyruk Gerçekleştirmi

Dizi

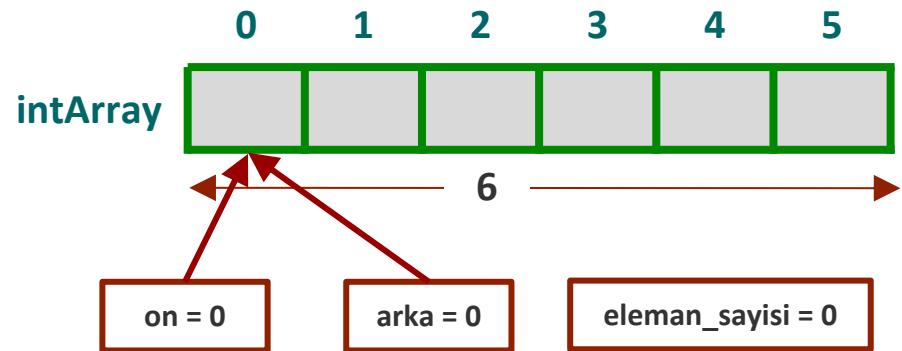


Baglantılı  
Listeler



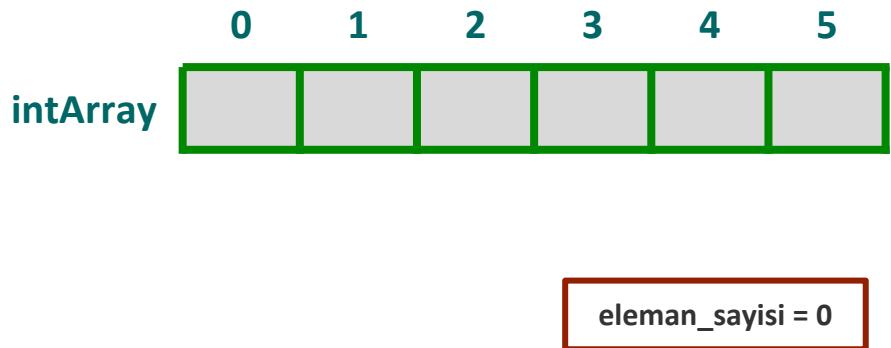
# Kuyruk Gerçekleştirmi

```
#define kapasite 6  
  
int intArray[kapasite];  
int on = 0;  
int arka = 0;  
int eleman_sayisi=0;
```



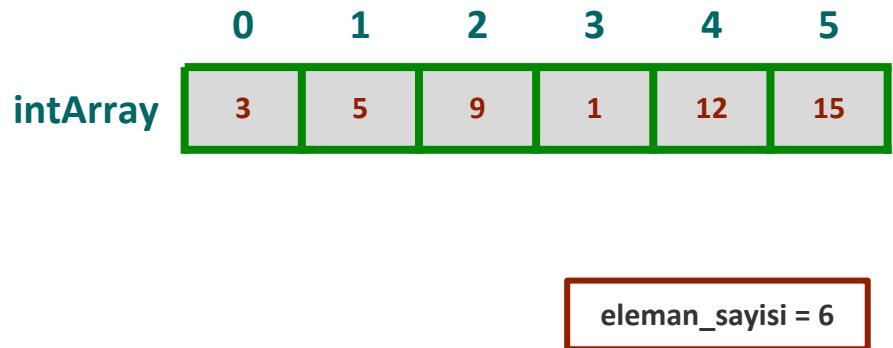
# Kuyruk Gerçekleştirmi

```
bool bos_mu(){  
    return eleman_sayisi==0;  
}
```



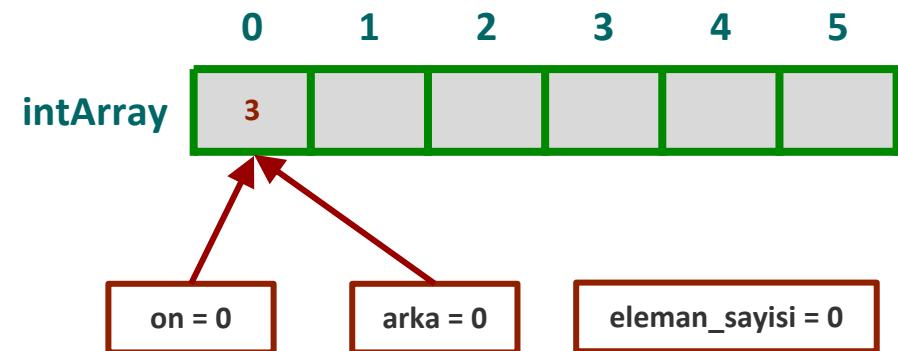
# Kuyruk Gerçekleştirmi

```
bool dolumu(){  
    return eleman_sayisi==kapasite;  
}
```



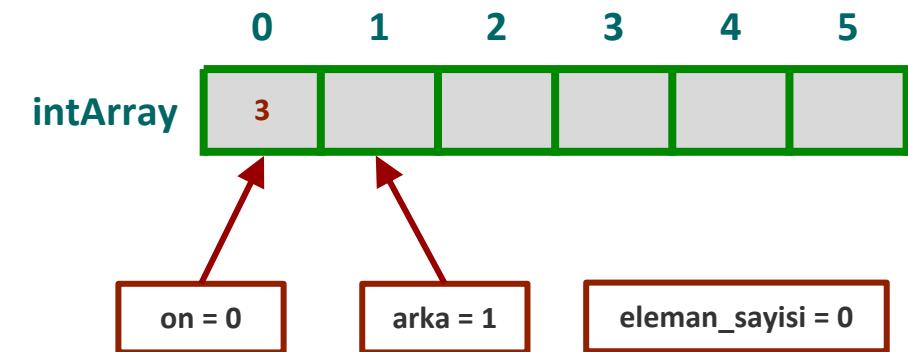
# Kuyruk Gerçekleştirmi

```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```



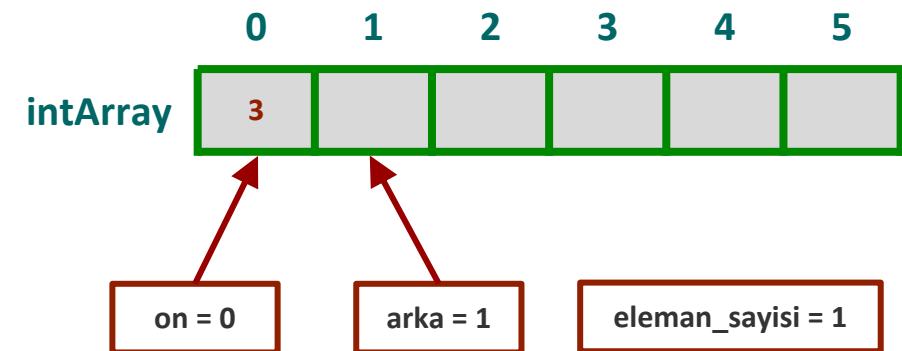
# Kuyruk Gerçekleştirmi

```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```



# Kuyruk Gerçekleştirmi

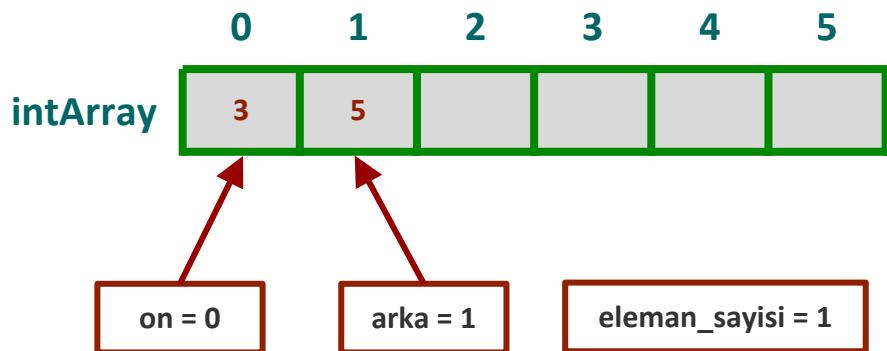
```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```



# Kuyruk Gerçekleştirmi

```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```

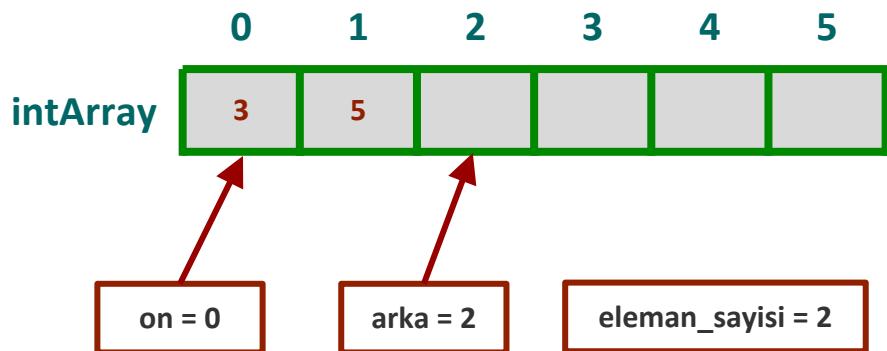
```
ekle(3);  
ekle(5);
```



# Kuyruk Gerçekleştirmi

```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```

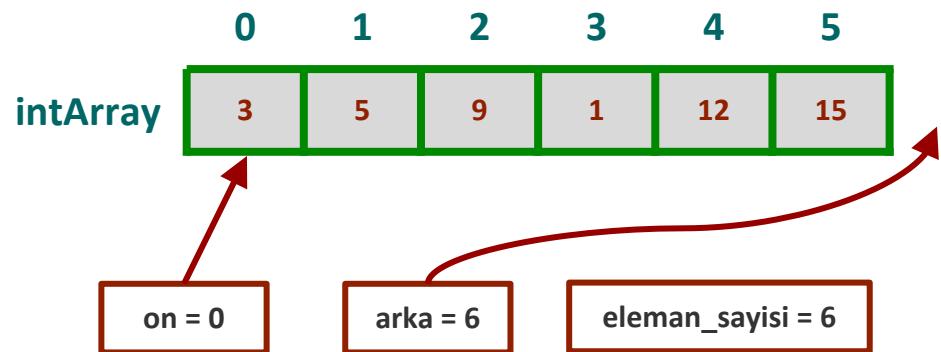
```
ekle(3);  
ekle(5);
```



# Kuyruk Gerçekleştirmi

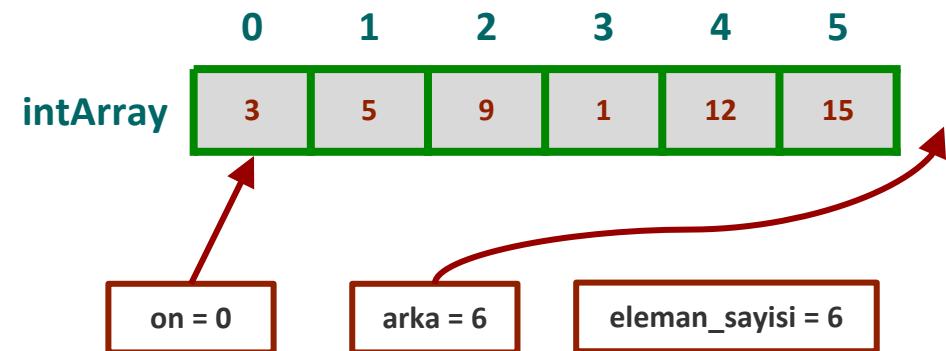
```
int ekle(int veri){  
    if(dolumu()) {  
        printf("Kuyruk dolu!\n");  
        return 0;  
    }  
    intArray[arka]= veri;  
    arka++;  
    if(arka==kapasite) arka=0;  
    eleman_sayisi++;  
    return 1;  
}
```

```
ekle(3);  
ekle(5);  
ekle(9);  
ekle(1);  
ekle(12);  
ekle(15);
```



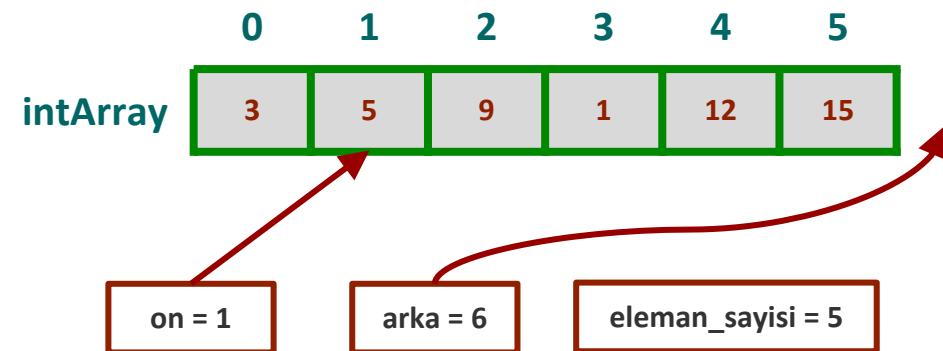
# Kuyruk Gerçekleştirmi

```
int cikar(){  
    if(bos_mu()){  
        printf("Kuyruk boş!");  
        return 0;  
    }  
    int data = intArray[on];  
    on++;  
    if(on==kapasite) on=0;  
    eleman_sayisi--;  
    return data;  
}
```



# Kuyruk Gerçekleştirmi

```
int cikar(){  
    if(bos_mu()){  
        printf("Kuyruk boş!");  
        return 0;  
    }  
    int data = intArray[on];  
    on++;  
    if(on==kapasite) on=0;  
    eleman_sayisi--;  
    return data;  
}
```



# Kuyruk(Bagli Liste)

```
struct KuyrukDugumu {  
    int veri;  
    struct KuyrukDugumu *  
sonraki;  
};
```

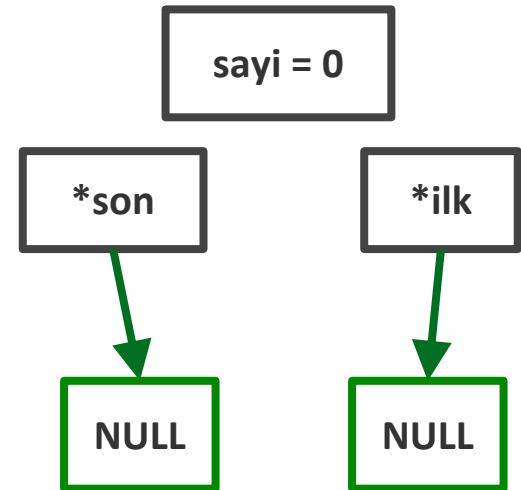
KuyrukDugumu



# Kuyruk(Bagli Liste)

```
class Kuyruk {  
private:  
    KuyrukDugumu * ilk;  
    KuyrukDugumu * son;  
    int _sayi;  
  
public:  
    Kuyruk() {  
        this->_sayi = 0;  
        this->ilk = this->son =  
NULL;  
    }  
    void ekle(int veri);  
    int al();  
    int sayi() { return this->_sayi; }  
};
```

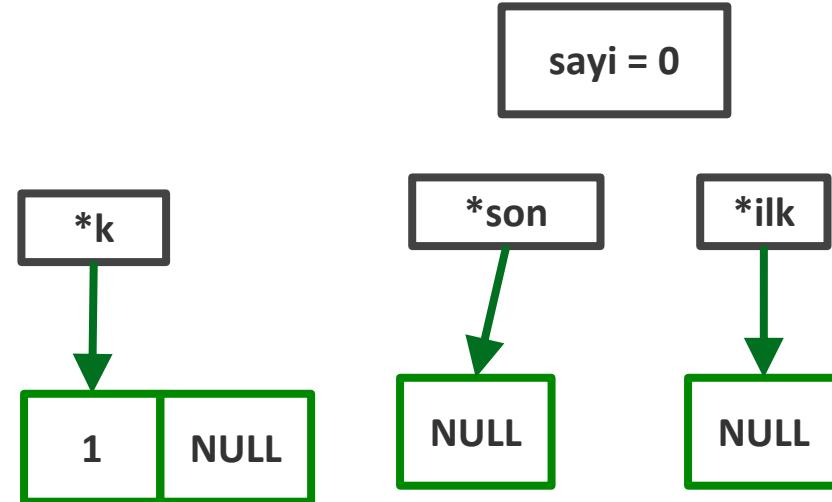
```
Kuyruk *k = new Kuyruk();
```



# Kuyruk(Bagli Liste)

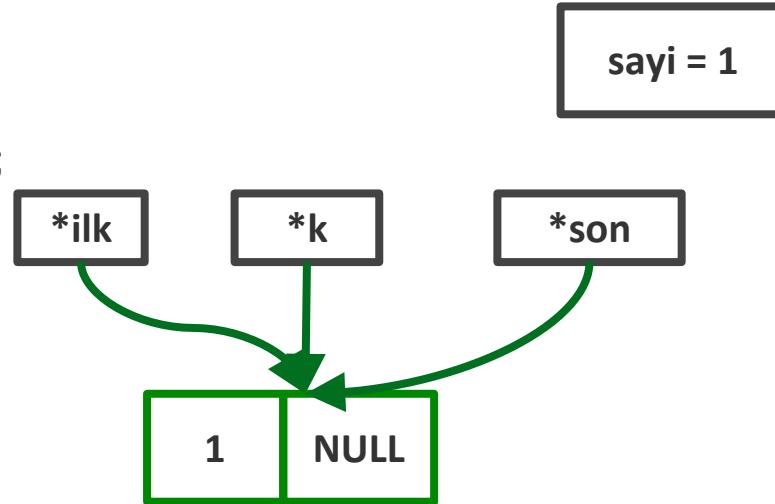
```
void Kuyruk::ekle(int veri) {  
    // bagli liste de sona ekleme islemi  
    KuyrukDugumu *k = new  
    KuyrukDugumu();  
    k->veri = veri;  
    k->sonraki = NULL;  
  
    if (this->ilk == NULL) {  
        this->ilk = this->son = k;  
    } else {  
        this->son->sonraki = k;  
        this->son = k;  
    }  
  
    this->_sayi++;  
}
```

```
k->ekle(1);
```



# Kuyruk(Bagli Liste)

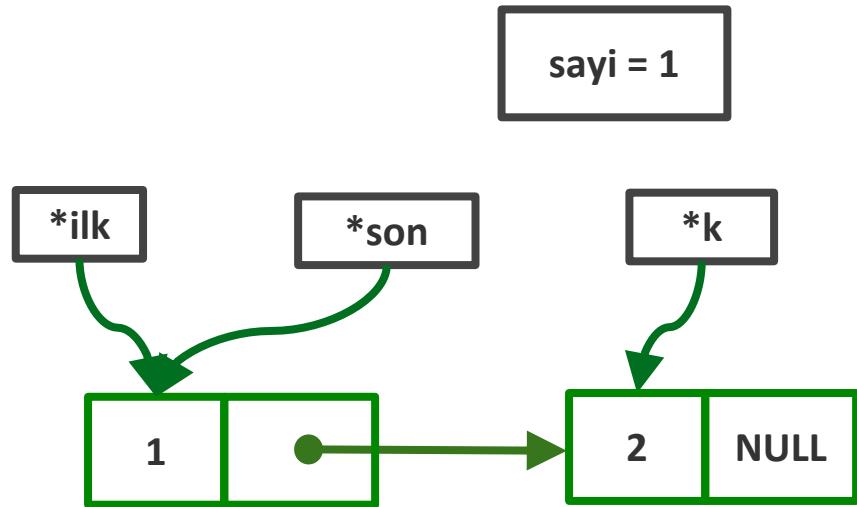
```
void Kuyruk::ekle(int veri) {  
    // bagli liste de sona ekleme islemi  
    KuyrukDugumu *k = new  
    KuyrukDugumu();  
    k->veri = veri;  
    k->sonraki = NULL;  
  
    if (this->ilk == NULL) {  
        this->ilk = this->son = k;  
    } else {  
        this->son->sonraki = k;  
        this->son = k;  
    }  
  
    this->_sayi++;  
}  
  
    k->ekle(1);
```



# Kuyruk(Bagli Liste)

```
void Kuyruk::ekle(int veri) {  
    // bagli liste de sona eklemeye islemi  
    KuyrukDugumu *k = new  
    KuyrukDugumu();  
    k->veri = veri;  
    k->sonraki = NULL;  
  
    if (this->ilk == NULL) {  
        this->ilk = this->son = k;  
    } else {  
        this->son->sonraki = k;  
        this->son = k;  
    }  
  
    this->_sayi++;  
}
```

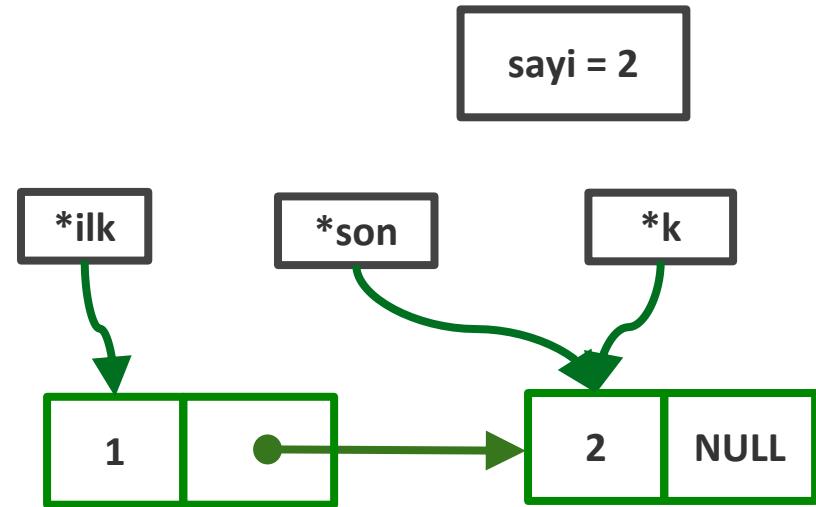
```
k->ekle(1);  
k->ekle(2);
```



# Kuyruk(Bagli Liste)

```
void Kuyruk::ekle(int veri) {  
    // bagli liste de sona eklemeye islemi  
    KuyrukDugumu *k = new  
    KuyrukDugumu();  
    k->veri = veri;  
    k->sonraki = NULL;  
  
    if (this->ilk == NULL) {  
        this->ilk = this->son = k;  
    } else {  
        this->son->sonraki = k;  
        this->son = k;  
    }  
  
    this->_sayi++;  
}
```

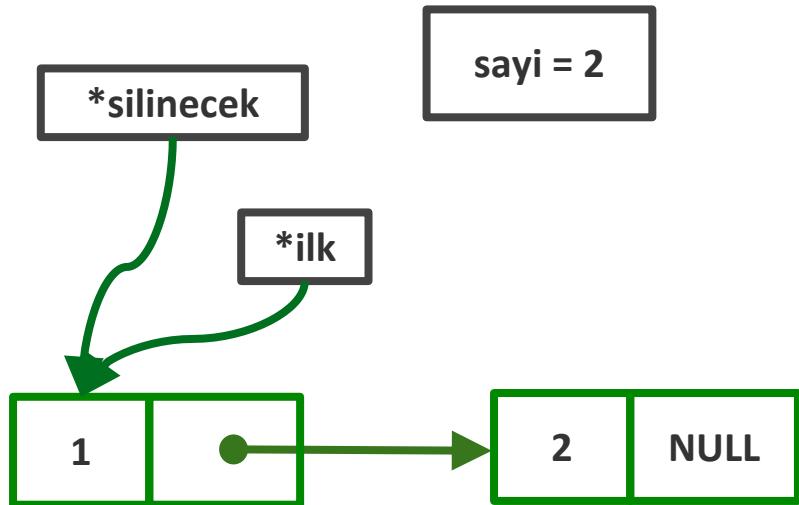
```
k->ekle(1);  
k->ekle(2);
```



# Kuyruk(Bagli Liste)

```
int Kuyruk::al( ) {  
    assert( this->sayi() != 0 ); // eleman yoksa hata ver  
    int veri = this->ilk->veri;  
    // bagli listede bastan silme islemi  
    KuyrukDugumu * silinecek = this->ilk;  
    this->ilk = this->ilk->sonraki;  
    delete silinecek;  
  
    this->_sayi--;  
    return veri;  
}
```

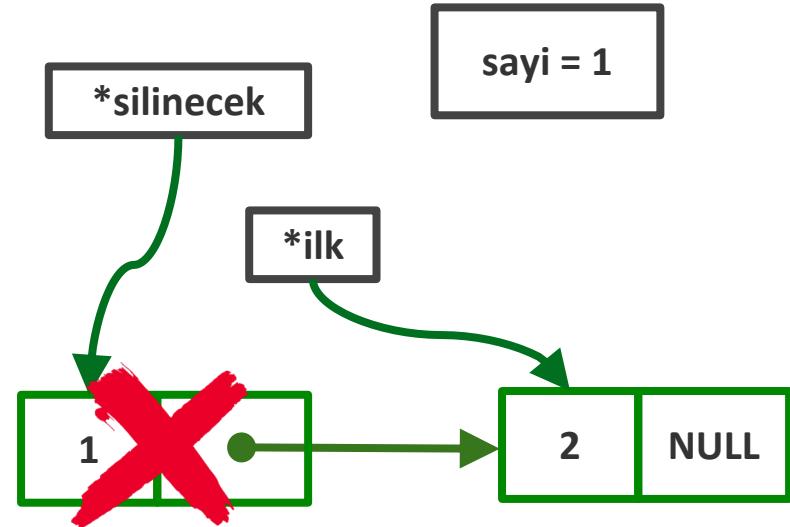
k->al( );



# Kuyruk(Bağlı Liste)

```
int Kuyruk::al( ) {  
    assert( this->sayi() != 0 ); // eleman yoksa hata ver  
    int veri = this->ilk->veri;  
    // bagli listede bastan silme islemi  
    KuyrukDugumu * silinecek = this->ilk;  
    this->ilk = this->ilk->sonraki;  
    delete silinecek;  
  
    this->_sayi--;  
    return veri;  
}
```

k->al( );



# Sorular

