YIĞINLAR

- Yığın İşlemleri
- Postfix, Prefix, Infix

Yrd.Doç.Dr. M. Ali Akcayol G. Ü. Bilgisayar Mühendisliği Bölümü

YIĞINLAR

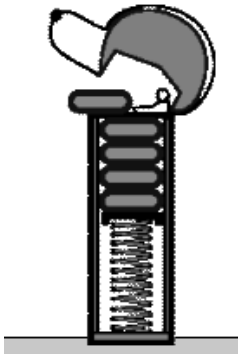




G. Ü. Bilgisayar Mühendisliği Bölümü

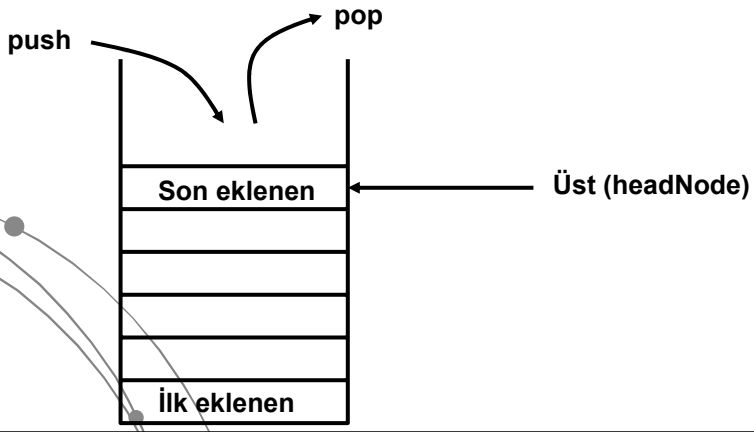
YIĞINLAR

- **Yığındaki elemanlardan sadece en son eklenene erişim yapılır.**
- **Yığına ilk eklenen eleman en son elde edilir.**
- **FILO (First-in-Last-out) veya**
  LIFO (Last-in-First-out) mantığıyla çalışır.
- **İki tane temel işlem yapılabilir ;**
  - **push, yığının sonuna yeni bir eleman ekleme**
  - **pop, yığının en üstündeki elemanın alınması**



YIĞINLAR

- **Dizilerle veya bağlı listelerle yapılabilir.**
- **Dizilerde boyut değiştirme ve yeni elemen ekleme zorluğundan dolayı genellikle bağlı dizilerle yapılır.**
- **En üst veya en son elemanı gösteren bir node tanımlanır (headNode).**



YIĞINLAR

Örnek kullanım yerleri :

- Yazılım uygulamalarındaki **Undo** işlemleri stack ile yapılır.

  Undo işlemi için LIFO yapısı kullanılır.

- Web browser'lardaki **Back** butonu (önceki sayfaya) stack kullanır.

  Buradada LIFO yapısı kullanılır

- Matematiksel işlemlerdeki operatörler (+,*,/,- gibi) ve operandlar için stack kullanılabilir.

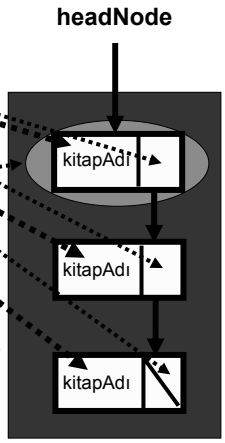
- Yazım kontrolündeki parantezlerin kontrolünde stack kullanılabilir.

YIĞINLAR

Stack oluşturma :

```
class stackNodeC
{
    public string kitapAdi;
    public stackNodeC sonraki;
    public stackNodeC(string kitapAdi)
    {
       this.kitapAdi = kitapAdi;
    }
}
class stackC
{
    public stackNodeC headNode;
    public stackC(string kitapAdi)
    {
       this.headNode = new stackNodeC(kitapAdi);
       this.headNode.sonraki = headNode;
    }
}

stackC kitapYigin = new stackC("");
```

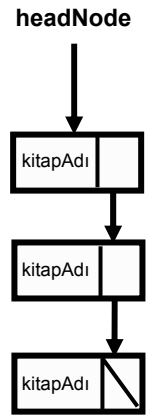


YIĞINLAR

Stack işlemleri :

- **boş yığın** **stackSize() == 0**
- **eleman sayısı** **stackSize()**
- **eleman ekleme** **push(kitapAdi)**
- **eleman alma** **pop()**

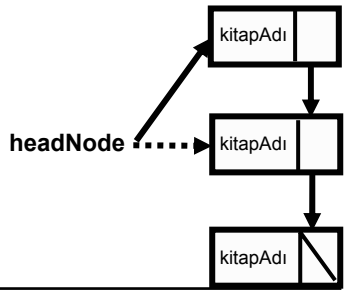
```
public int stackSize()
{
    stackNodeC aktif = new stackNodeC("");
    aktif = kitapYigin.headNode;
    int i = 0;
    while (aktif.sonraki != aktif)
    {
      aktif = aktif.sonraki;
      i++;
    }
    return i;
}
```



YIĞINLAR

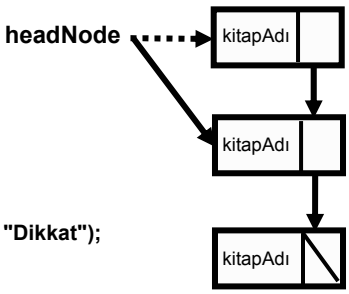
Stack işlemleri (eleman ekleme)

```
public void push(string kitapAdi)
{
    if (stackSize() >= MaxSize)
      MessageBox.Show("Yığın maksimum elemana sahip ! Yeni
      eleman eklenemez !", "Dikkat");
    else
    {
      stackNodeC yeniNode = new stackNodeC(tBKitapAdi.Text);
      yeniNode.sonraki = kitapYigin.headNode;
      kitapYigin.headNode = yeniNode;
      lStackSize.Text = "Stack Size =
      "+Convert.ToString(stackSize());
    }
}
```



YIĞINLAR

Stack işlemleri (eleman alma)




```
public void pop()
{
    if (stackSize() == 0)
    {
      MessageBox.Show("Yığında eleman yok !", "Dikkat");
    }
    else
    {
      kitapYigin.headNode = kitapYigin.headNode.sonraki;
    }
}
```


YIĞINLAR

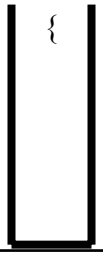
Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).

İşlem sırası

{…[…(…)…]…}

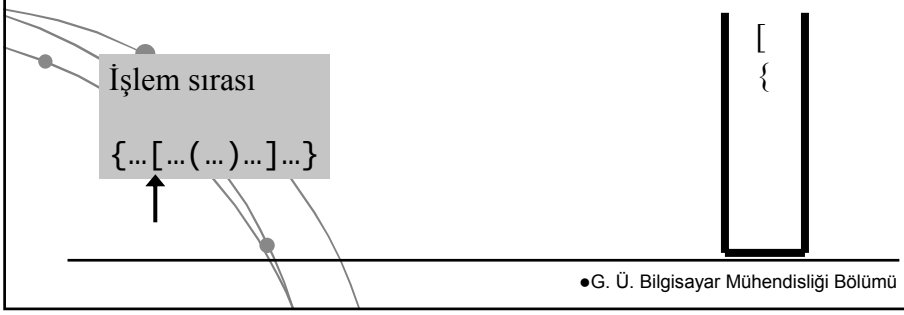


YIĞINLAR

Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).

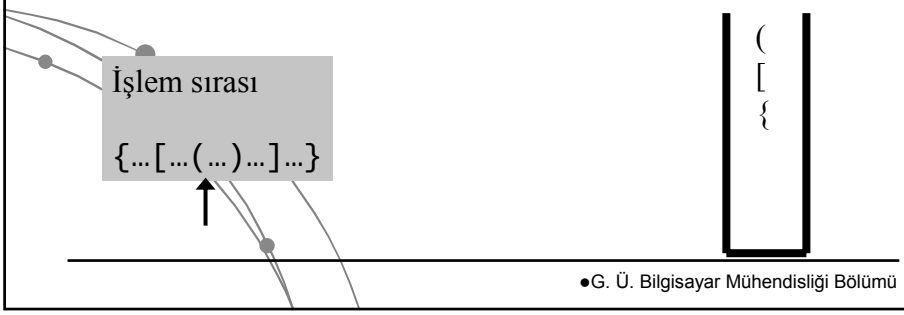


YIĞINLAR

Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).

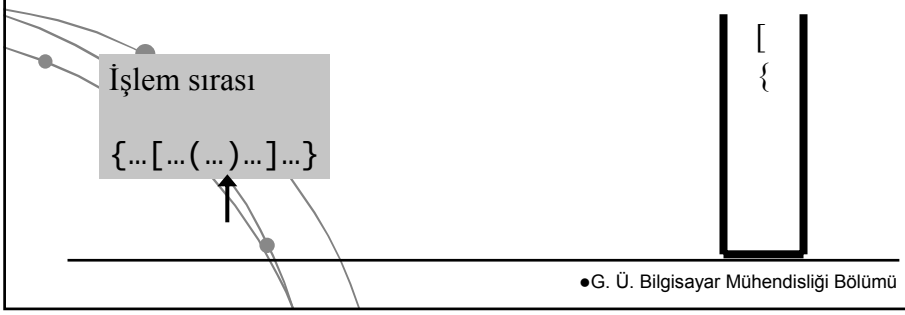


YIĞINLAR

Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).

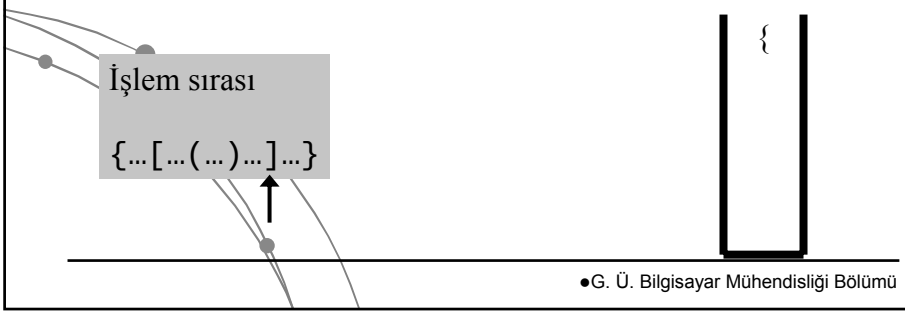


YIĞINLAR

Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).

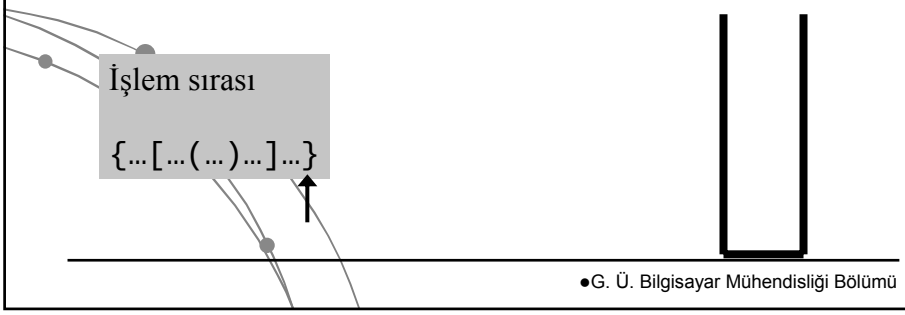


YIĞINLAR

Örnek:

Parantez unutma hatalarını bulma */, },) or].

Normal yazım şekli —[( )],
Hatalı yazım şekli [( ]).



YIĞINLAR

İşlem:

(a+{b-c}*[d+(a+b))

Stack:

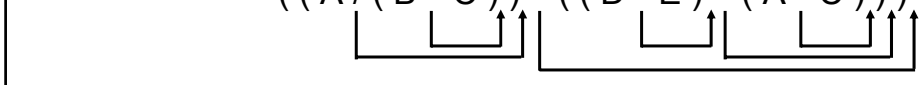
- (
- ({
- (
- ([
- ([(
- ([

Hata !

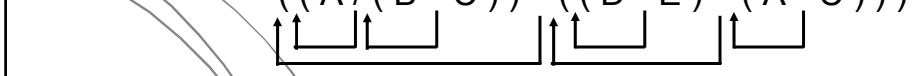
YIĞINLAR

Infix, Postfix, Prefix:

Infix ((A / (B ^ C)) – ((D * E) – (A * C)))



Postfix A B C ^ / D E * A C * – –



Prefix – / A ^ B C – * D E * A C

YIĞINLAR

Postfix Örnek:

Aşağıdaki infix gösterimini postfix haline dönüştürünüz.

Infix A * B + (C – D / E) postfix AB*CDE/-+

ch	opstack	postfix	açıklama	ch	opstack	postfix	açıklama
	#		push #		+	AB*	pop * –>postfix, push ch
A			read ch		#		
		A	ch –> postfix	(			read ch
*			read ch		(		push ch
	*		push ch		+		
	#				#		
B			read ch	C			read ch
		AB	ch –> postfix			AB*C	ch –> postfix
+			read ch	-			read ch

YIĞINLAR

Postfix Örnek (devam):

Infix **A * B + (C – D / E)**

ch	opstack	postfix	açıklama	ch	opstack	postfix	açıklama
	-		push ch		(		
	(				+		
	+				#		
	#			E			read ch
D			read ch			AB*CDE	ch -> postfix
		AB*CD	ch –> postfix	)			read ch
/			read ch			AB*CDE/-	(paranteze kadar pop
	/		push ch	+			read ch
	-					AB*CDE/-+	ch -> postfix
				#		AB*CDE/-+#	pop # -> postfix

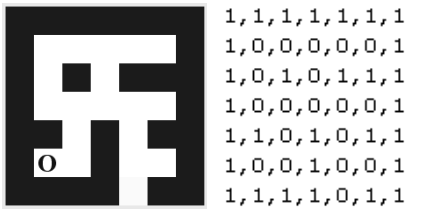
YIĞINLAR

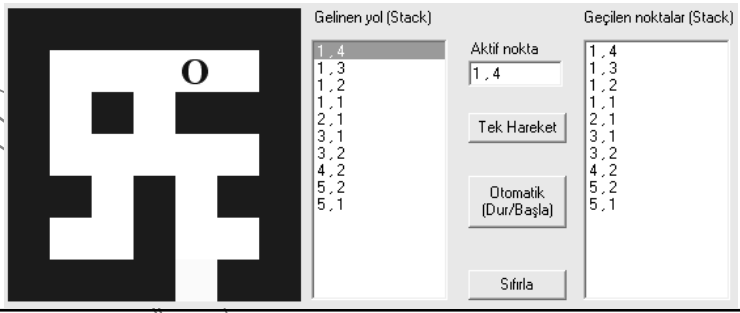
Postfix Kurallar ve İşlemler:

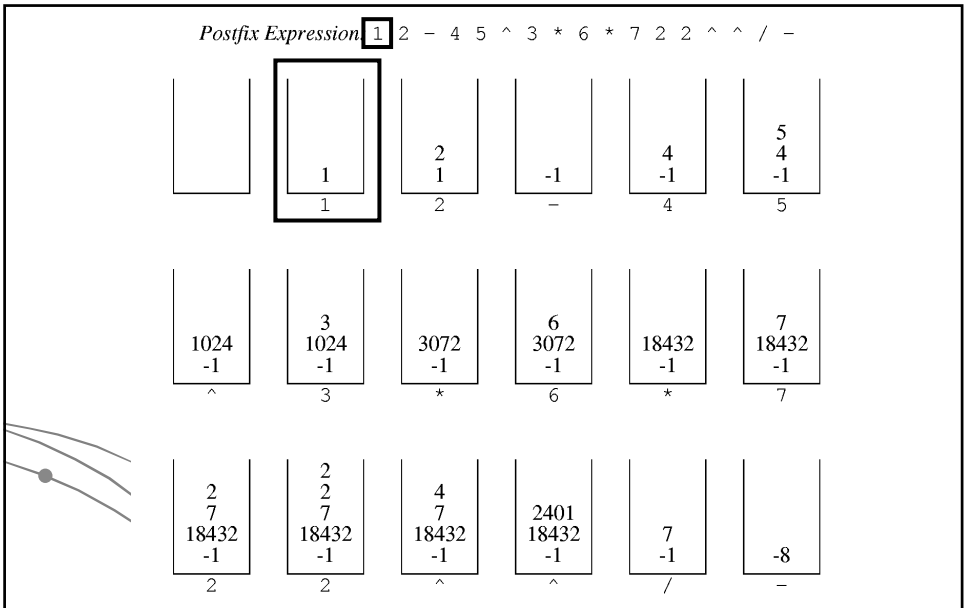
1. **opstack yığınını # işareti ekleyerek başlat**
2. **infix içinden bir sonraki karakteri (ch) oku**
   - **Eğer ch operand ise postfix'e ekle**
   - **Eğer ch sağ parantez ise opstack'ta sol parantezi görene kadar tüm operatörleri pop yap ve postfix'e ekle**
   - **Eğer ch bir operatör ise opstack'taki önceki operatörleri # işaretini görene kadar pop yap ve postfix'e ekle**
3. **Opstack'taki son # işaretini görene kadar 1. ve 2. adımları tekrar et.**

YIĞINLAR

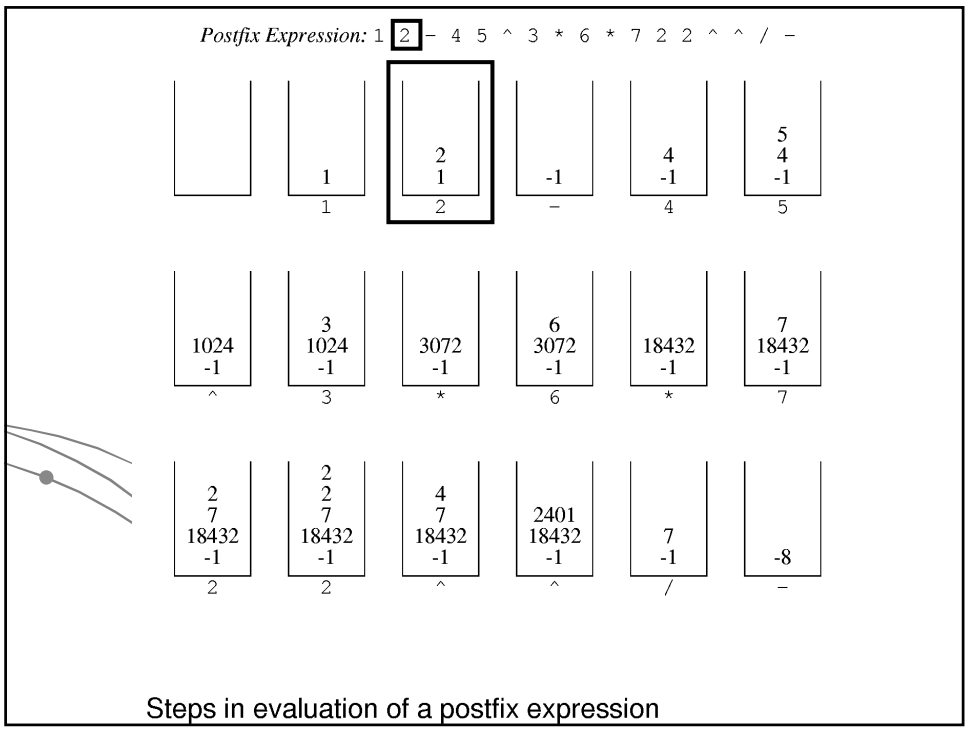
Örnek Labirent Oyunu (Maze): Backtraking uygulaması



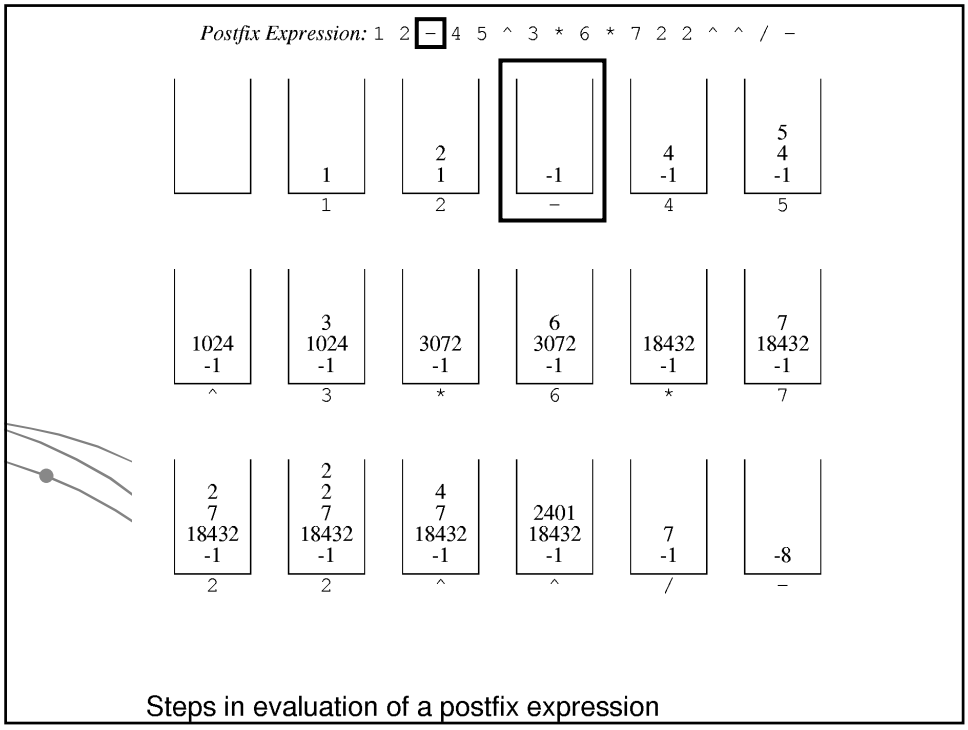




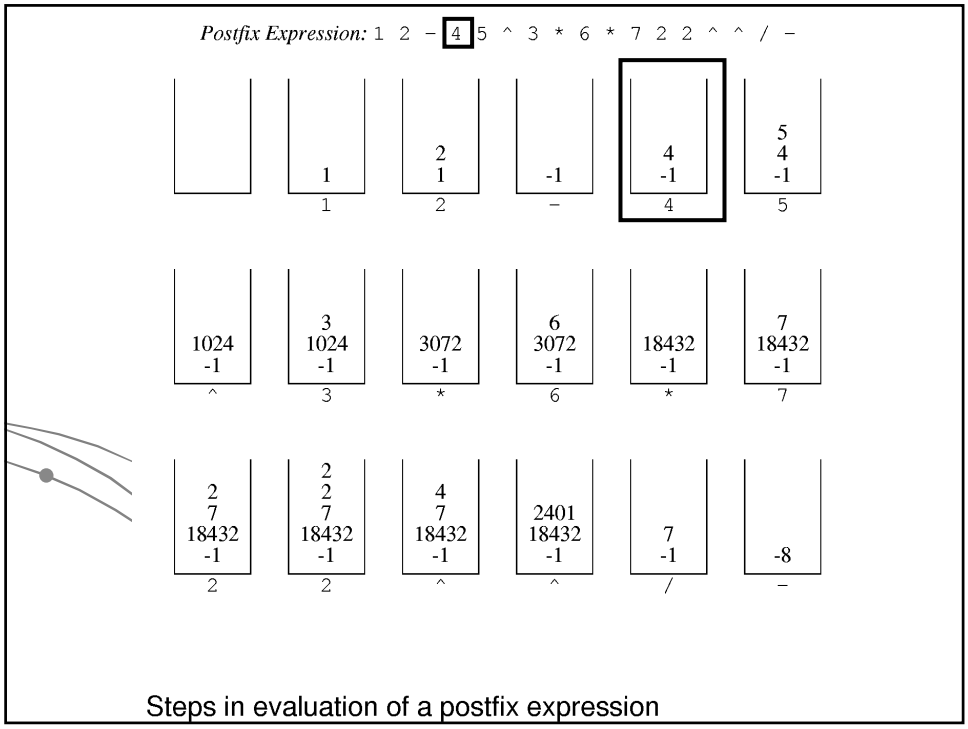
Steps in evaluation of a postfix expression



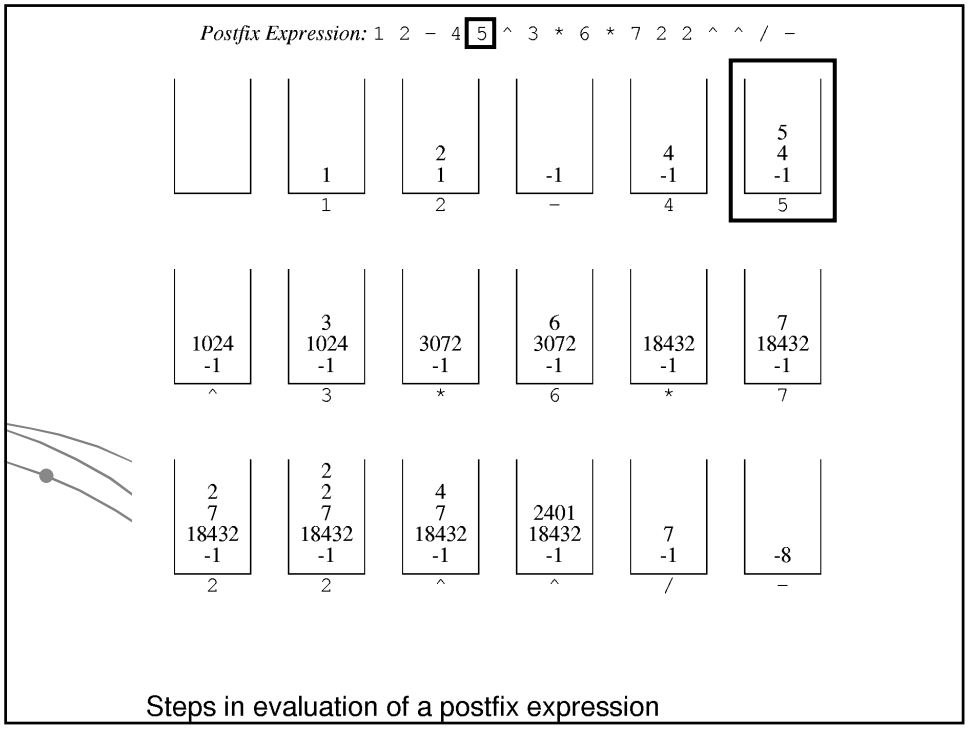
Steps in evaluation of a postfix expression



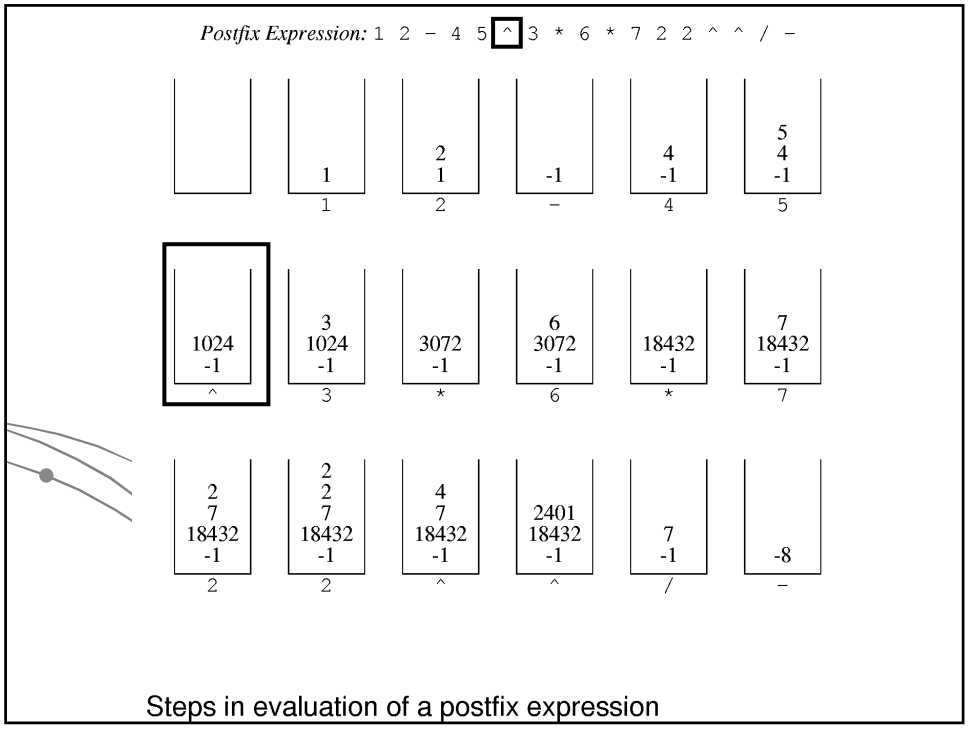
Steps in evaluation of a postfix expression



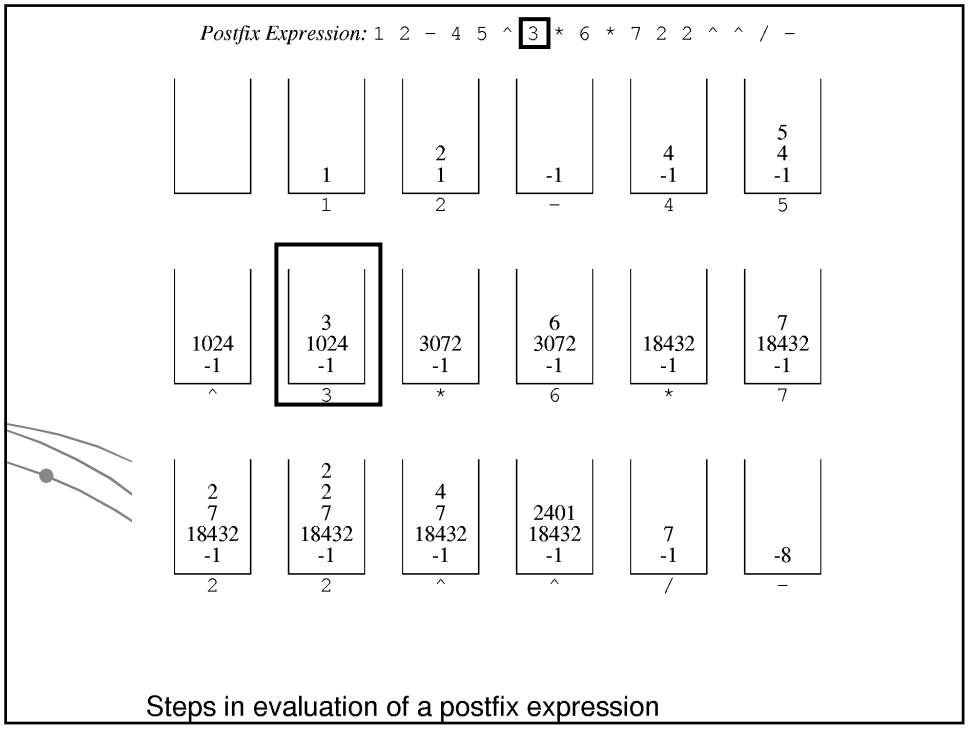
Steps in evaluation of a postfix expression



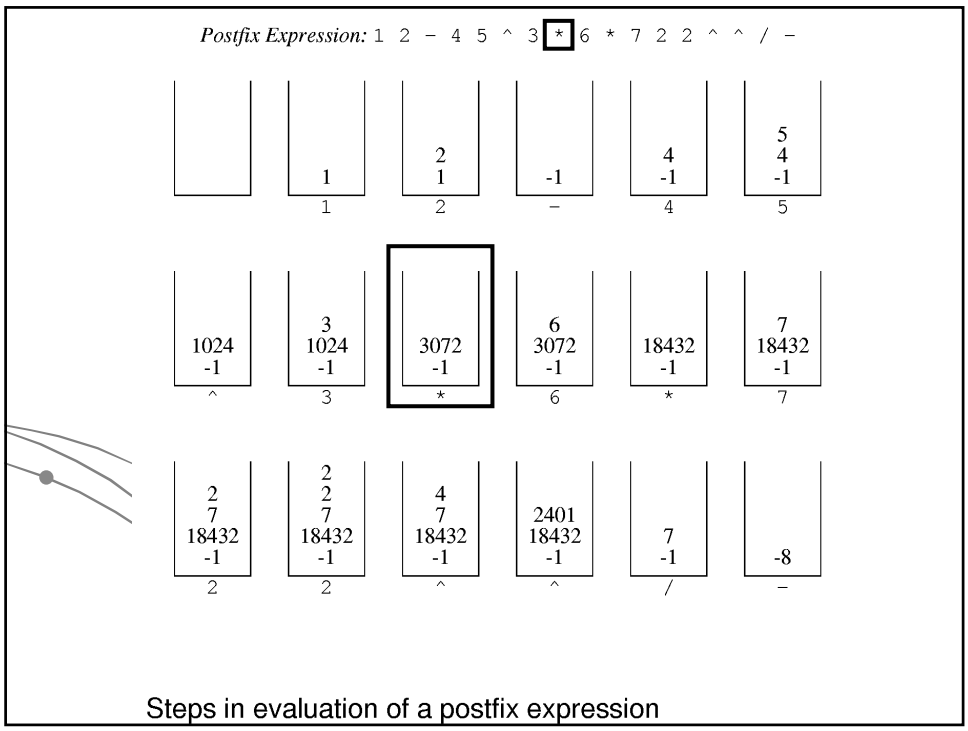
Steps in evaluation of a postfix expression



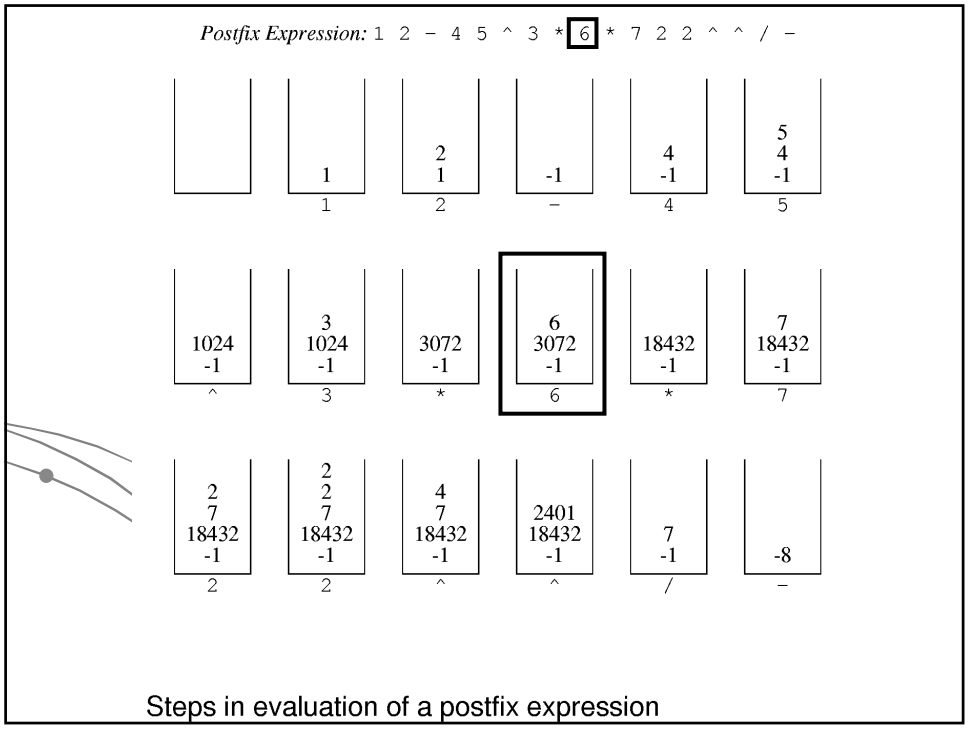
Steps in evaluation of a postfix expression



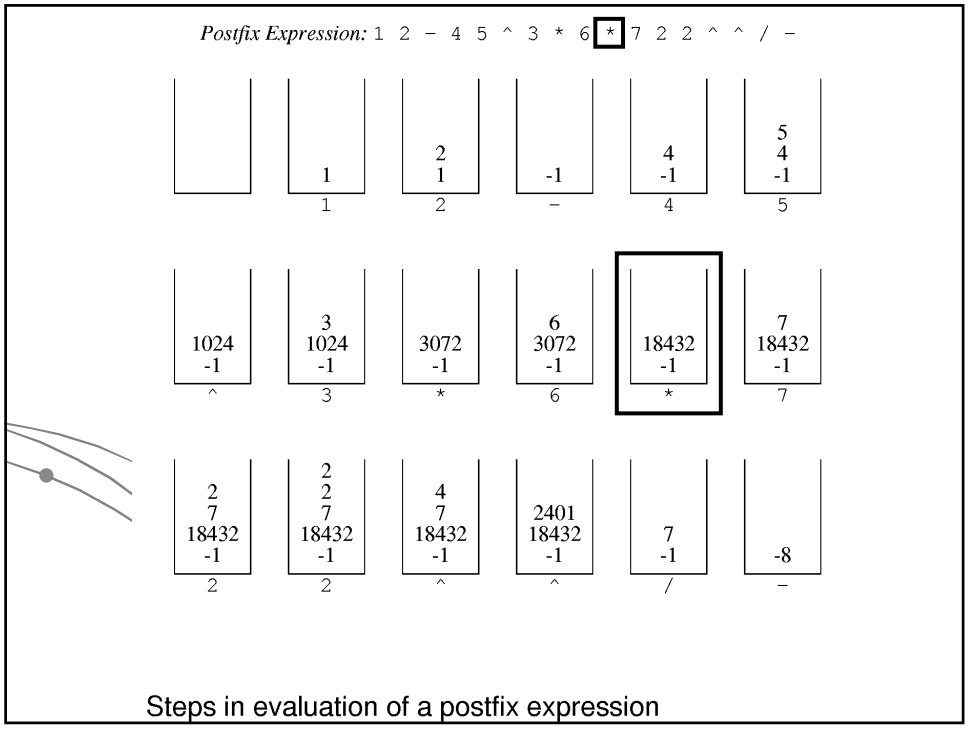
Steps in evaluation of a postfix expression



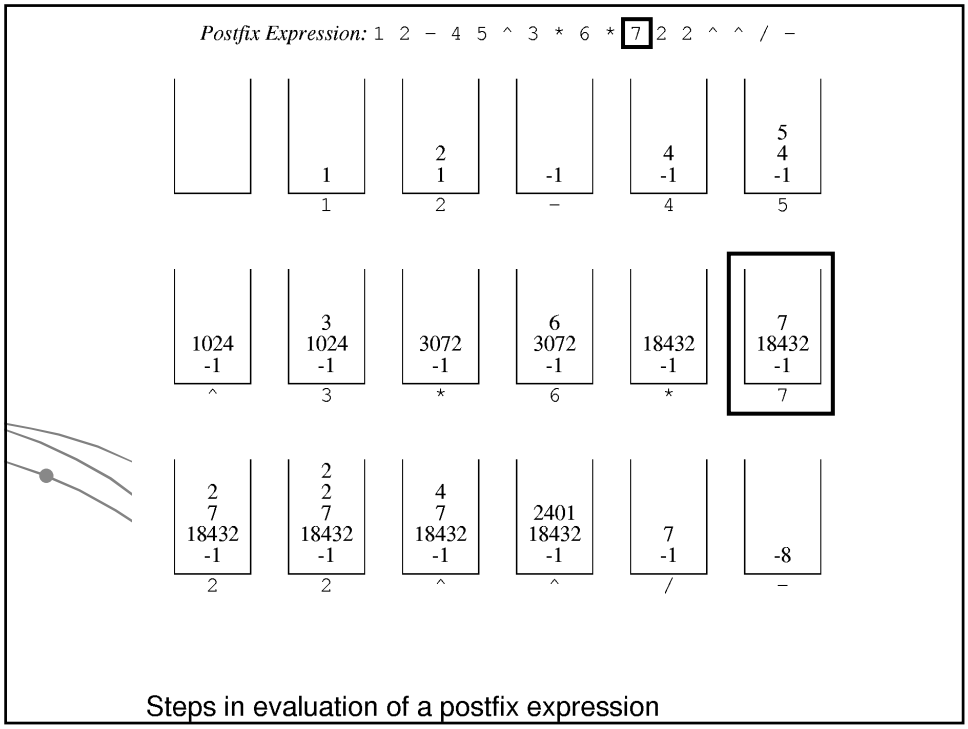
Steps in evaluation of a postfix expression



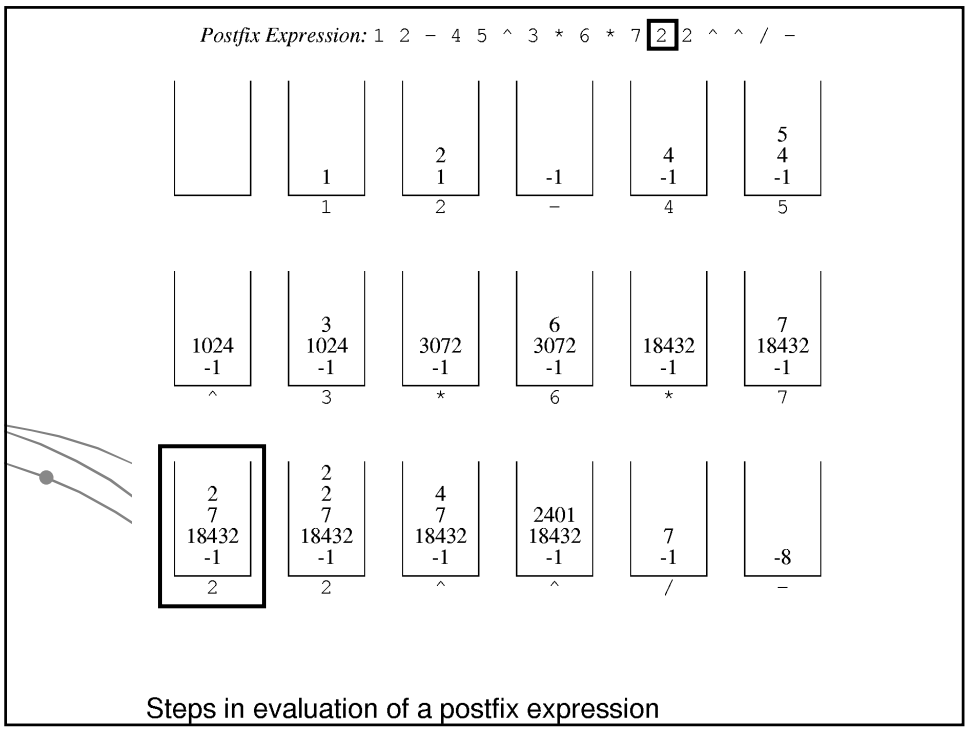
Steps in evaluation of a postfix expression



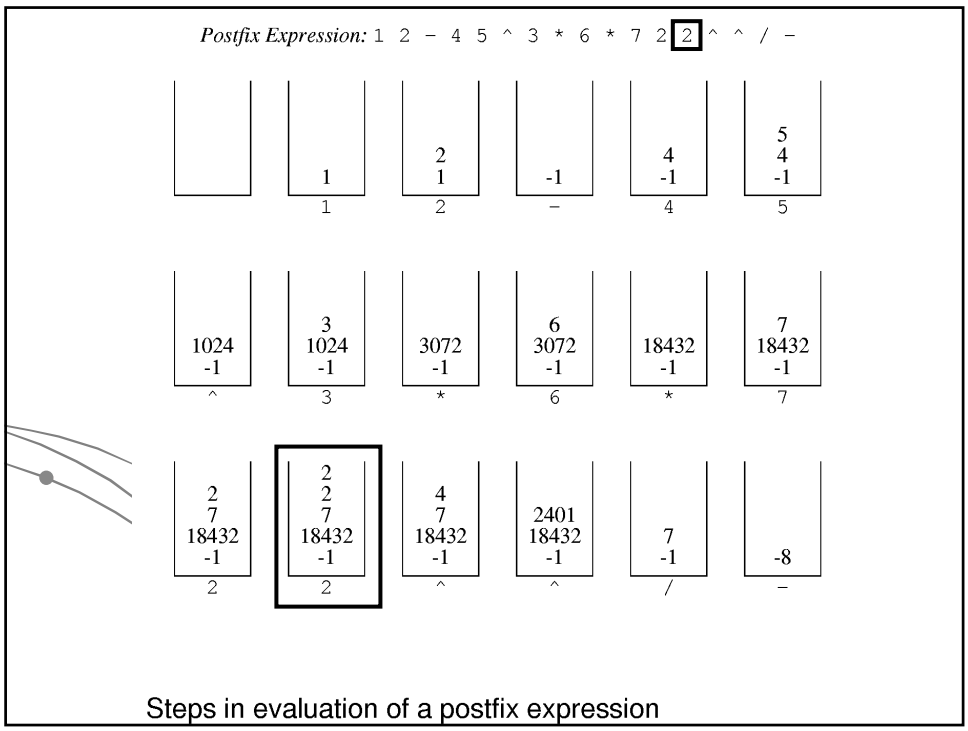
Steps in evaluation of a postfix expression



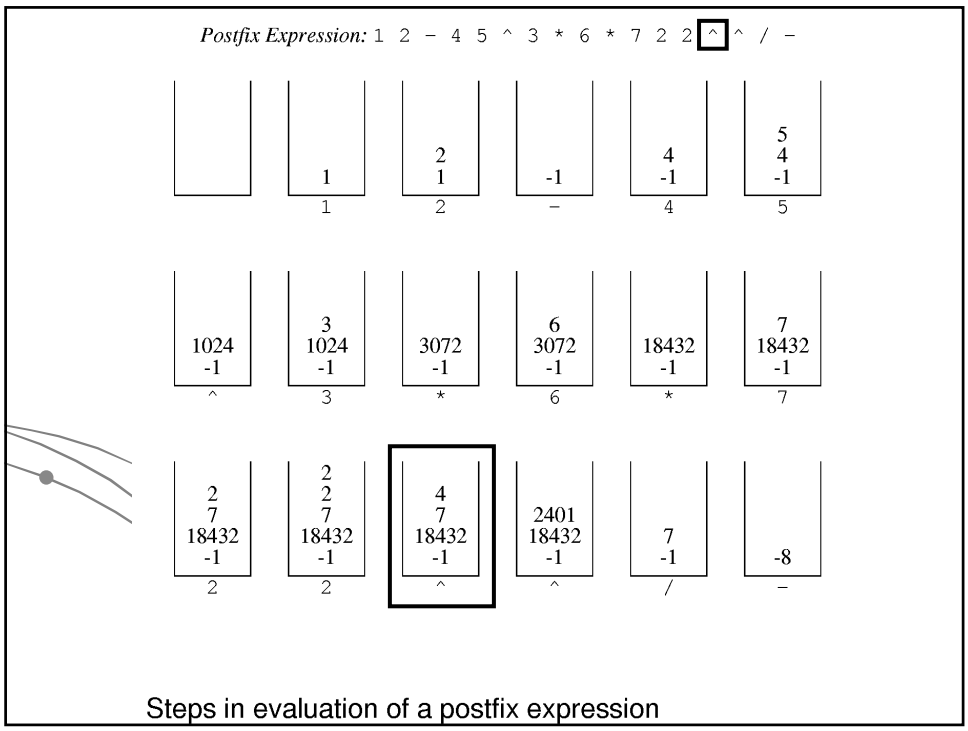
Steps in evaluation of a postfix expression



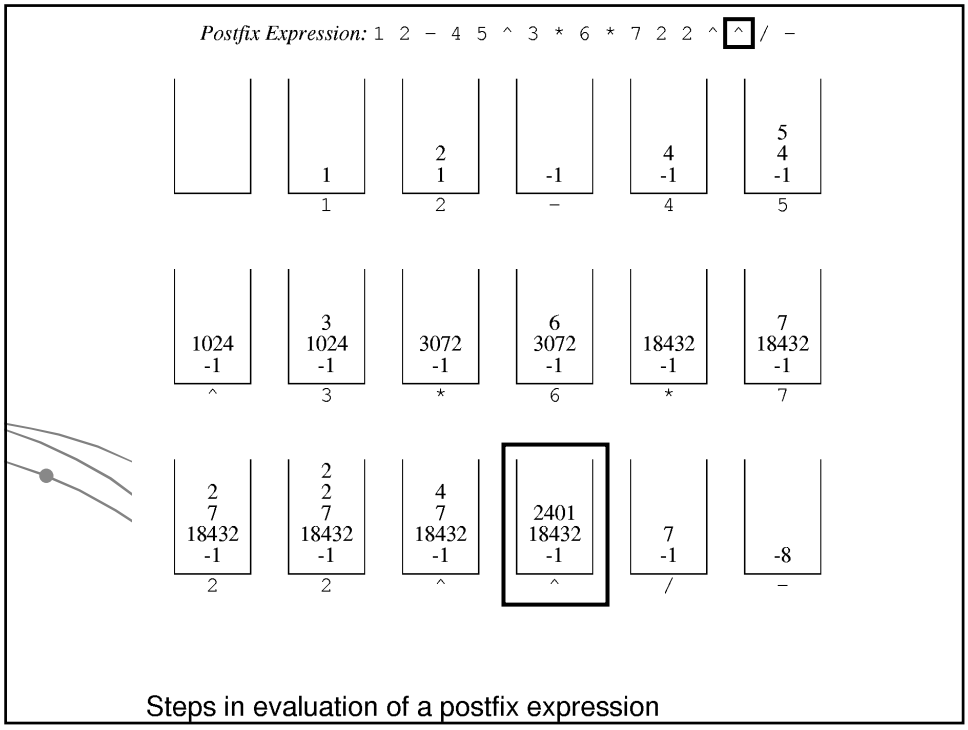
Steps in evaluation of a postfix expression



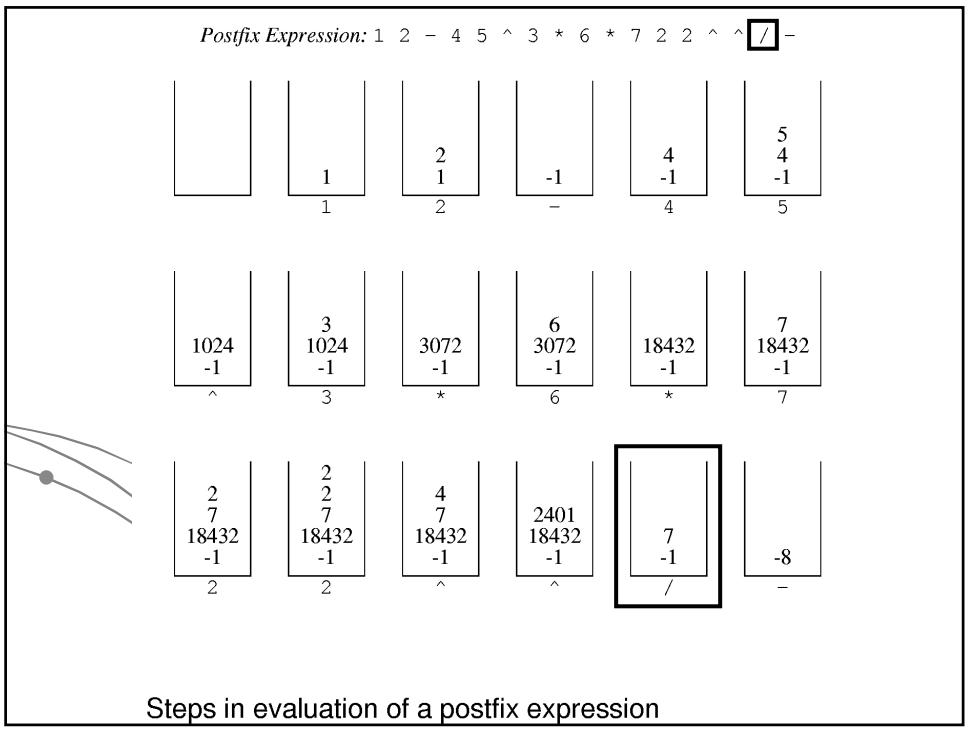
Steps in evaluation of a postfix expression



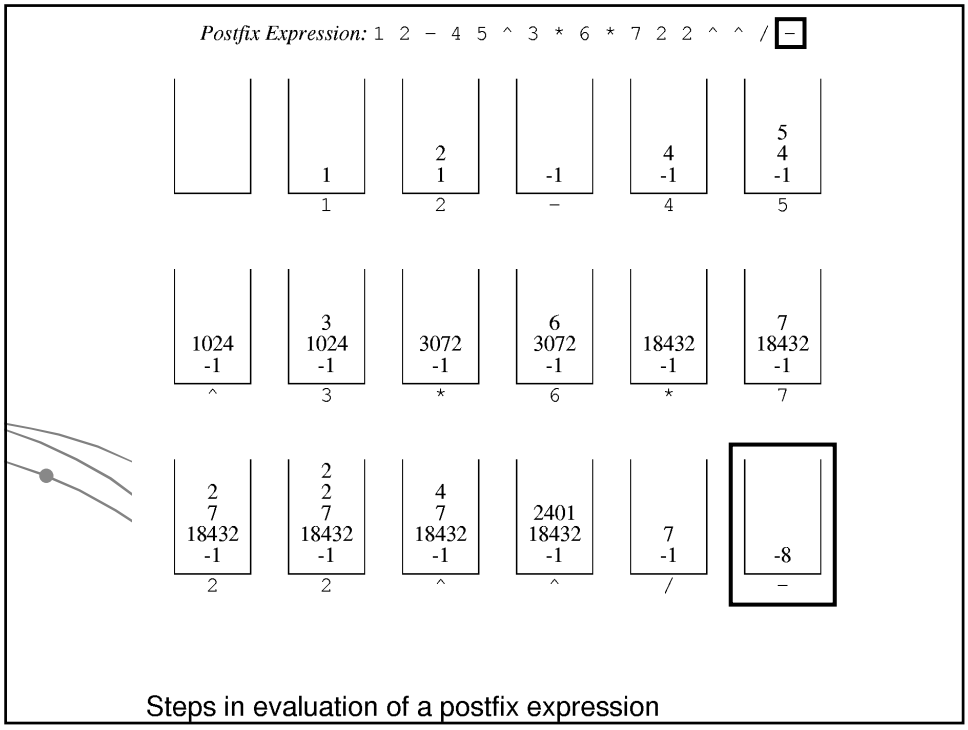
Steps in evaluation of a postfix expression



Steps in evaluation of a postfix expression



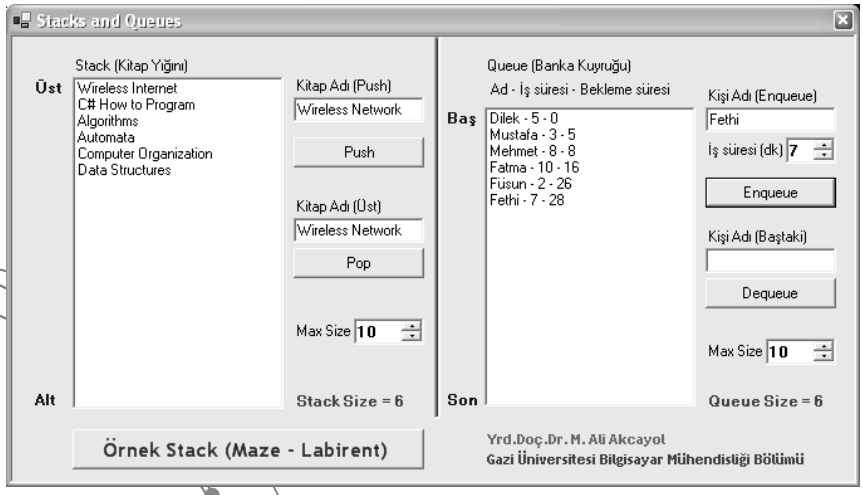
Steps in evaluation of a postfix expression



Steps in evaluation of a postfix expression

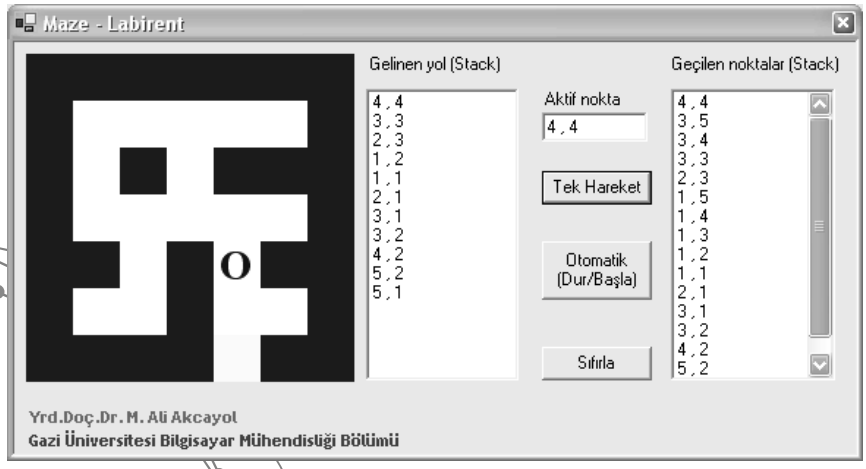
YIĞINLAR

Uygulama programı (Stacks ve Queues):



YIĞINLAR

Uygulama programı (Maze - Stack):

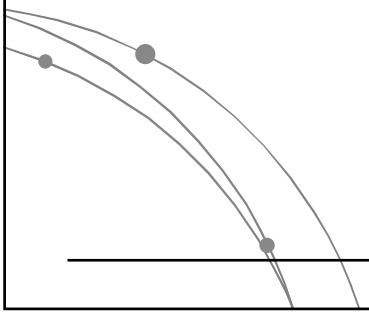


YIĞINLAR

Haftalık Ödev:

1- Örnek programı verilen Maze (Labirent) uygulamasını 12x12 boyutunda yeniden düzenleyiniz. Başlangıç olarak istediğimiz noktadan başlayabilmemizi sağlayınız.

2- postfix olarak yazılmış ifadeyi hesaplayarak sonucu bulan bir program yazınız. Örnek olarak 5 3 2 - / 4 7 * + ifadesinin sonucunu bulunuz. Sonuç = 33 olarak bulunacak.



YIĞINLAR

Gelecek Hafta

Kuyruklar

(Queues)