Örnek(Parantez Eşleme Örneği) "Yigin_c.cpp"

Matematiksel bir ifadede açma ve kapama parantezlerini kontrol eden:

1- Açma parantezi eksikse: (2+3)) -> Açma Parantezi eksik
2- Kapama Parantezi Eksikse: ((2+3)->Kapama Parantezi Eksik
3- Eksiklik Yoksa: (2+3)->Parantez Hatası Yok
   Çıktılarını üretecek ve yığın yapısı yardımıyla fonksiyonları yazalım.

Değişen kod parçaları (karakter tutulacağından dolayı)

```
typedef struct Yigin
      {   int indis;
          char eleman[YBoyut];
      }Yiginlar;


void Yigina_Ekle(char ekle)

char Yigindan_Cikar();

int Parantez_Kotrol(char *islem)
{ int i;
      for (i=0;i<strlen(islem);i++)
      {
            if (islem[i]=='(')
                  Yigina_Ekle(islem[i]);
            else
            {
                  if (islem[i]==')')
                        {if(Yigin_Bosmu()==-1)
                              return -1;
                        else
                              Yigindan_Cikar();
                        }
            }

      }
      if (Yigin_Bosmu(Yeni_Yigin)==-1) return 0;
            else
            return 1;



}
```

```
void main()
{
      Yiginlar *Yeni_Yigin=Yigin_Olustur();
      int i;
      char secim;
      char *islem;//="(3+4)/5)";
      clrscr();
      printf("Kontrol stringi=>");
      scanf("%s",islem);
      i=Parantez_Kotrol(islem);
      switch(i)
      {
      case -1:printf("Hata:Acma Parantezi Eksik");
                  break;
      case 0:printf("Parantez Hatasi Yok");break;
      case 1:printf("Hata:Kapama parantezi Eksik");


      }

}
```

Infix den Postfix'e Dönüşüm:

Infıx:a+b : operatör operandların arasında.

Postfix:ab+ : operator operantlardan sonra.

Prefix:+ab : operator operantlardan önce.

Infix notasyonun bir dezavantajı operatörlerin değerlendirme kontrolu için parantezler kullanılmasıdır. Postfix ve prefix notasyonlarda parantezler kullanılmaz. Infix notasyonu kullanan yüksek seviyeli dillerde ifadeler direkt değerlendirilemezler. En genel değerlendirme tekniği infix notasyonun postfix notasyona çevrilerek değerlendirilmesidir.

Kurallar:

Yalnızca operatörler yığına eklenir.

1- Yığına Gönderilen > yığın içindeki ise yığına eklenir, <= ise yığından operatör çıkarılır.
2- Kapama parantezi ile karşılaşıldığında açma parantezine kadar olan tüm yığındaki ifadeler yığından çıkarılarak sonuca yazılır.
3- Tüm ifadeler işlendiğinde (İşlem bittiğinde) yığın boş değilse yığında kalan tüm operatörler yığından çıkarılarak sonuca yazılır.

Simge	Yığın içindeki Öncelik değeri	Yığına girerken sahip olduğu öncelik değeri
)	-	-
*, /	2	2
+, -	1	1
(	0	4

Örnek: A+B*C-D/E

Infix	Yiğin	Postfix
A+B*C-D/E		
+ B*C-D/E		A
B*C-D/E	+	A
*C-D/E	+	AB
C-D/E	* +	AB
-D/E	* +	ABC
-D/E	+	ABC*
-D/E		ABC*+
D/E	-	ABC*+
/E	-	ABC*+D
E	/ -	ABC*+D
	/ -	ABC/+DE
	-	ABC/+DE/
		ABC*+DE/-

Örnek: A*(B+C)*D

Infix	Yiğin	Postfix
A*(B+C)*D		
*(B+C)*D		A
(B+C)*D	*	A
B+C)*D	(*	A
+C)*D	(*	AB
C)*D	+ (*	AB
)*D	+ (*	ABC
*D	(*	ABC+
*D	*	ABC+
*D		ABC+*
D	*	ABC+*
	*	ABC+*D
		ABC+*D*

Postfix ifadelerin değerlendirilmesi:

Algoritması : değerlendirme algoritması.

```
Algoritma posfix_degerlendir (değer ifade <string>)
İfade_boyutu=string boyutu
Yeni_Yığın=Yığın_Oluştur();
İndex=0;
Loop (index<İfade_Boyutu)
      if (ifade[index] operand)
            Yigina_Ekle(Yeni_Yığın,ifade[index]);
      Else
            operand2=Yigindan_Cikar(Yeni_Yığın);
            operand1=Yigindan_Cikar(Yeni_Yığın);
            operator=ifade[index];
            değer=hesapla(operand1,operator,operand2);
            Yığına_Ekle(Yeni_Yığın,değer);
      End if;
      index=index+1;
end loop;
      sonuç=Yığından_Çıkar(Yeni_Yığın);
      return sonuç;
end Algoritma;
```

örnek: 2 3 5 - * postfix ifedesinin sonucu nedir?