T.C
GAZİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BMT-210
VERİ YAPILARI VE ALGORİTMALAR

ÖDEV-5

AHMET CELİL YALMAN
22181616702

DOÇ.DR. ADEM TEKEREK

Sorular ve cevapları

1- Verilen bir karakter dizisindeki parantezlerin eşli olup olmadığını yığın (stack) yapısını kullanarak belirleyen fonksiyonu bağlı liste mantığına göre yazınız.

Ödev için java dili ile birlikte yığın veri yapısı ve bağlı liste veri yapısı kullanılmıştır. Bu dili kullanmamdaki neden string türünde pirimitiv olmayan değişkenleri fonksiyonlara parametre olarak göndermenin c++ gibi daha alt seviye dillere nazaran kolay olmasıdır.

Node veri yapısı:

```
4 usages
static class Node {  // node yapısı (tutulacak dataların türlerin ve pointerı) tanımlanır
    2 usages
    char data; // her bir nodun daya kısmı her bir parantez için
    3 usages
    Node next; // node a ait pointer

    1 usage
    public Node(char data) {
        this.data = data;  // structın data kısmı için atam işlemi yapılıyor
        this.next = null;  // başlangıçta node pointer herhangi biryeri işaret etmez
    }
}
```

Yığında kullanılacak node sistemi yukarıdaki yorum satırında belirtildiği gibi tanımlanmıştır. İlgili node sisteminde bulunan data kısmı için char türünde değişken kullanılmıştır. İlgili nodun pointerı “next” olarak tanımlanır. Java saf bir OOP olması nedeniyle ilgili class sistemi için yapıcı fonksiyon kullanılır.

Stack veri yapısı:

```
2 usages
static class Stack { // yigin veri yapisi kur
    7 usages
    Node top;    // yığını en üst kısmı
    1 usage
    int indis = 1;  // node un indisi

    1 usage
    public Stack() {
        this.top = null;
        indis++;
    }

    1 usage
    public void push(char data) {  // listeye nodelar ekle
        Node newNode = new Node(data);
        newNode.next = top;
        top = newNode;
    }

    3 usages
    public boolean isEmpty() { return top == null; }

    1 usage
    public char pop() {
        if (isEmpty()) {
            return '0';
        } else {
            char poppedItem = top.data;
            top = top.next;
            return poppedItem;
        }
    }
}
```

Yığın veri yapısı tanımlanarak yığının en üst noktasını işaret eden “top” isimli değişken tanımlanır. Daha sonra yığın veri yapısı için indis tanımlaması yapılır. Bu sayede yığına eleman eklenirken veya elaman çıkartılırken ilgili konum bilinebilir. Yığın veri yapısı için push(ekleme) ve pop() isimli fonksiyonlar tanımlanmaktadır. Buna ilave olarak “isEmpty” adında yardımcı fonksiyon da bulunur ve bu yardımcı fonksiyon ile yığın veri yapısının en üst elemanını işaret eden pointerı (next isimli) null a atamasını gerçekleştirir.

Kontrol fonksiyonu(isBalanced):

```
1 usage
public static boolean isBalanced(String str) {
    Stack stack = new Stack();
    for (int i = 0; i < str.length(); i++) { // girilen parantez dizisinin boyutuna kadar tara

        char currentChar = str.charAt(i); //tek tek parantez işaretleri alınır

        if (currentChar == '(' || currentChar == '{' || currentChar == '[') {    // alınan parantezlerin tipi
            stack.push(currentChar);       // eklencek eleman parantez açılımı ile başlıyorsa push la

        } else if (currentChar == ')' || currentChar == '}' || currentChar == ']') {

            if (stack.isEmpty()) {return false;} // eklenecek eleman parantez kapatma ile başlarsa 0 döndür(false)

            else {   // eklenen eleman parantez kapama ise ve dizinin ilk elemanı  değilse
                char poppedItem = stack.pop();   // en üstteki elemanın alttakilerle eşleniği var mı diye bakılır

                if (currentChar == ')' && poppedItem != '(') {
                    return false;
                } else if (currentChar == '}' && poppedItem != '{') {
                    return false;
                } else if (currentChar == ']' && poppedItem != '[') {
                    return false;
                }
            }
        }
    }
    return stack.isEmpty(); // en üstteki eleman boş
}
```

Ödevde istenilen parantez türleri belirli olduğundan dolayı tasarlanacak fonksiyon ona göre üretilmiştir. İsterlerde, girilen parantez türlerinin eşleniği kontrol edilmesi gerekmektedir. Bunun için parantez açma işaretlerinden biri kontrol edilir. Bahsi geçen işaretlerden biri varsa fonksiyonun başında oluşturulan yığının push fonksiyonuna ilgili veri gönderilir. Eğer işaretlerden birisi parantez kapatma işareti ise false değeri döndürülür.

Ana fonksiyon(main):

```java
public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    while(true) { // birden fazla kontrol için

        System.out.print("parantez stringi giriniz... ");
        String parantezString = scanner.nextLine();   // parantez stringi al örn:([]{})[]


        if (isBalanced(parantezString)) {  // alınan stringi eşli olup olmadığını kontrol et
            System.out.println("parantez eşlidir");
        } else {
            System.out.println("parantez eşli değil");
        }
    }
}
```

Parantez stringi alınırak “isBalanced” fonksiyonuna parametre olarak gönderilir. Yukarıda açıklanan fonksiyonun içeriğinde true döndürülmesiyle yığında bulunan parantez string i eşli diğer durumlarda false döndürüleceği için parantez eşli olmamıştır. Bu durumlar konsola bastırılmıştır.

Örnekler:

```
C:\Users\celil\.jdks\openjdk-20\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.1\lib\idea_rt.jar=52725:C:\Program Files\JetBrains\Intel
parantez stringi giriniz... ()[]{}{}()[]
parantez eşlidir
parantez stringi giriniz... {()[{()}]}
parantez eşlidir
parantez stringi giriniz... ()[(
parantez eşli değil
parantez stringi giriniz... {}({{}}
parantez eşli değil
parantez stringi giriniz...
```

Sabit dizi ilse stack veri yapısı zaman karmaşıklığı:

işlem	Zaman karmaşıklığı
Push(ekleme)	O(1)
Pop(silme)	O(1)
Top(üstten eleman çekme)	O(1)

Bağlı liste ile stack veri yapısı zaman karmaşıklığı:

İşlem	Zaman karmaşıklığı
Push(ekleme)	O(1)
Pop(silme)	O(1)
Top(üstten eleman çekme)	O(1)

Alan karmaşıklığı tablosu:

Dizi/liste	Alan karmaşıklığı
Sabit dizi(statik)	O(n)
Bağlı liste(dinamik)	O(n)

Sabit dizi, bellekte belirli bir boyutta bir dizi oluşturarak stack elemanlarını saklar. Ekleme ve silme işlemleri O(1) zaman karmaşıklığına sahiptir, ancak dizi boyutu önceden belirlendiği için doluluk kontrolü için ayrı bir fonksiyon gereklidir. Ayrıca, dizi boyutunun değiştirilememesi dezavantaj olarak görülebilir.

Bağlı liste, stack elemanlarını bağlı düğümler olarak saklar. Ekleme ve silme işlemleri yine O(1) zaman karmaşıklığına sahiptir ve boyutu önceden belirlenmemiş olduğu için dinamik olarak büyüyebilir. Ancak, bellek yönetimi için ekstra işlem gücü gerektirir ve sabit diziye kıyasla daha fazla alan karmaşıklığına sahiptir.

Özetle, sabit dizi stack yapısı bellek yönetimi için daha az kaynak gerektirse de boyutu önceden belirlenmiş olduğu için esneklik sınırlıdır. Bağlı liste ise daha esnek olmasına rağmen daha fazla bellek yönetimi gerektirir. Seçim, kullanım senaryosuna ve önceliklere bağlı olarak değişebilir.