**Bil212 Veri Yapıları**

**Lab2**

**Bu labda iki kısım vardır.**

1. *MyStack*  sınıfını built-in **Queue** kullanarak implement etmeniz gerekmektedir. MyStack, stack sınıfına ait metodları içermekle birlikte bazı özelleşmiş metodları da içermektedir. Aşağıda metodları görebilirsiniz.

**Metotlar:**

***push( ):*** parametre olarak aldığı değeri stack in en üstüne ekler

***pop( ):*** stack in en üstündeki değeri döner ve stack ten çıkarır

***peek( ):*** stack in en üstündeki değeri döner

***search( ):*** parametre olarak aldığı değerin indeksini döner. Tekrar eden eleman ise en son eklenenin indeksini döner.

***findMultiple( ):*** frekansı en yüksek olan elemanı döner. Frekansları eşit birden çok eleman varsa en son stack e ekleneni döner.

***reverse( ):*** stack i ters çevirir

***toString( ):*** stack i, ilk eklenen elemandan son eklenene doğru arada boşluk olacak şekilde yazdırmalıdır.

******

Şekil1

Şekil1, karakter tutan bir MyStack nesnesinin iç görünümünü göstermektedir. Stack e ekleme yapılacağı taraf da görsele eklenmiştir. Aşağıda bazı metodların veri yapısında nasıl değişiklikler yaptığı ve hangi çıktıları verdiği gösterilecektir.

*search(j):*

döndüğü değer: 3

veri yapısında bir değişiklik yok.

*findMultiple( ):*

döndüğü değer: s

veri yapısında bir değişiklik yok

*reverse( ):*

değer dönmez

******

1. Tasarlanan MyStack sınıfını kullanarak input olarak verilen ve parantezlerden oluşan bir string deki en uzun anlamlı parantez ikilisinin indekslerini dönen IndexParantesis adlı metodu implement etmeniz gerekmektedir. Bu metodu *Parantesis* sınıfında implement etmelisiniz. Bazı durumlar ve metodun çıktısı aşağıda gösterilmiştir.

**Örnek1:**

) ( ( ) ( ( ) )

metodun döndüğü değer: 5, 8

**Örnek2:**

) ( ( ) ( ( ) ) )

metodun döndüğü değer: 2, 9

**Yüklenecekler:**

*MyStack.java ve Parantesis.java* adlı dosyaları *Surucu.java ile çalıştıktan sonra* UZAK a yükleyebilirsiniz.