VERİ YAPILARI VE ALGORİTMALAR PROJE RAPORU

Projenin Amacı : Kullanıcıdan alınan girdileri bağlantılı liste veri yapısında saklamak ve kullanıcının **undo**, **redo** ve **yaz** işlemlerini yapmasını sağlamaktır.

Undo : Kullanıcının girdiği son girdiyi listeden çıkarır.

Redo : Yapılan son undo işlemini geri alır.

Yaz : Listedeki elemanları listeye eklenme sırasına göre ekrana yazdırır.

Q : Programdan çıkış yapılmasını sağlar.

Programda, tek yönlü bağlantılı liste veri yapısı kullanılmıştır. List ve list2 olmak üzere iki adet liste kullanılmıştır. List listesi girilen verileri tutan liste, list2 ise undo işlemi ile çıkarılan verileri tutar.

Program Nasıl Çalışır ?

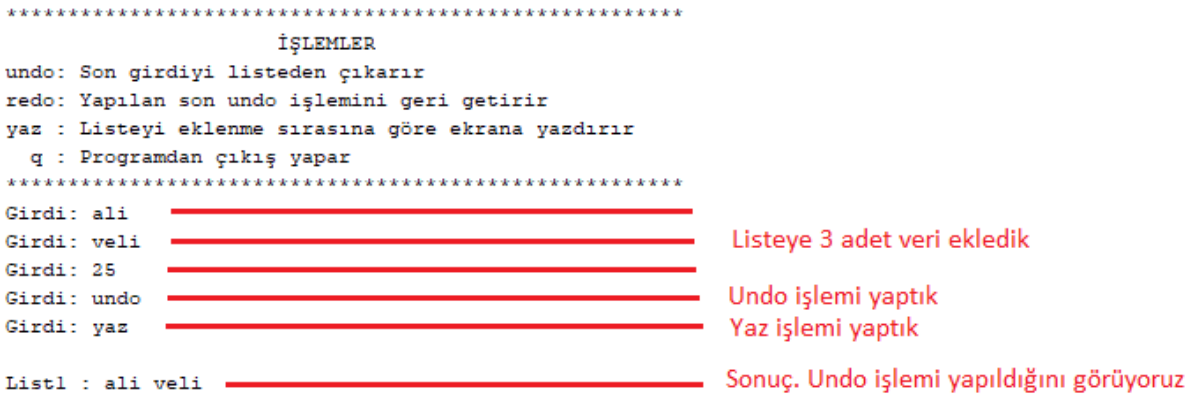
Program, çalıştırıldığında ekrana **undo**, **redo**, **yaz** ve **q** işlemlerini tanıtan bir açıklama yazısı yazmaktadır. Bilgilendirme yazısından sonra program kullanıcıdan girdi istemektedir.

Listeye Ekleme İşlemi

Kullanıcıdan alınan girdi **undo, redo, yaz** veya **q** değilse alınan girdi listeye eklenir.

Undo İşlemi

Program çalıştırıldıktan sonra kullanıcının girdiği ilk girdi undo ise listemiz boş olduğu için herhangi bir çıkarma işlemi yapamayacaktır. Program, listeden çıkarma işlemini yapamadığı için kullanıcıya **"Liste boş. Undo işlemi yapamazsınız!"** uyarısında bulunur. Listeye herhangi bir veri eklendikten sonra kullanıcıdan alınan girdi undo ise program listeye son eklenen elemanı listeden çıkartır ve list2 adlı listeye baştan ekler. Çıkartılan elemanın list2 listesine eklememizin sebebi ise undo işleminden sonra kullanıcı redo işlemi yaparsa list2 listesine eklenen veriyi tekrar list listesine geri gönderebilmek içindir. Eğer kullanıcı listedeki verilerin hepsini undo yaparsa ve tekrar undo işlemi yapmak isterse yine listenin boş olduğu durumdaki gibi **"Liste boş. Undo işlemi yapamazsınız!"** uyarısını alır.



Redo İşlemi

Program çalıştırıldıktan sonra kullanıcının girdiği ilk girdi redo ise öncesinde undo işlemi yapılmadığından dolayı program redo işlemi yapmaz ve kullanıcıya **"Undo işlemi yapılmadan redo yapılamaz!"** uyarısında bulunur. Program çalıştırıldıktan sonra kullanıcı listeye ekleme işlemleri, sonrasında ise undo işlemi yaptıktan sonra redo işlemi yapmak isterse program son yapılan undo işlemini geri getirir. Yani undo işleminde list listesinin sonundan silip list2 listesinin başına eklediği veriyi, list2 listesinin başından silip list listesinin sonuna ekler. Programımız da eğer eklenme işlemi yapılmış ise ve sonrasında redo işlemi yapılmak isteniyorsa eklenme işleminden önceki undo

işlemleri yok sayılır ve redo işlemi yapılmaz. Şimdi ise redo işleminde olabilecek bazı durumları inceleyelim.

1. Kullanıcı **undo – ekleme – yaz – redo** işlemlerini sırası ile yaptırmak ister ise program ne sonuç verir?

   Programımızda ekleme işleminden sonra, önceki undo işlemleri yok sayılır ve redo yapılamaz kuralı olduğundan dolayı son işlem olan redo işleminden önceki işlemin bilgisi bize lazımdır. Bir önceki işlemimiz **yaz** olduğu için bu redo işlemi yapmamıza engel değildir. Ancak yaz işleminden önceki işlem undo ise program redo yapabilir eğer undo değilse ve işlem ekleme işlemi ise program redo yapamaz. Şimdi bize redo işleminden iki önceki işlem lazım olacaktır. İki önceki işlemi bulmak yerine farklı bir yöntem kullandık. Program list listesine her ekleme yaptığında list2 listesindeki tüm verileri sildirdik. Böylece ekleme işleminden sonra kullanıcı redo yapmak ister ise list2 boş olduğundan dolayı redo işlemi yapılamaz ve kullanıcıya **"Listeye ekleme yapıldıktan sonra redo yapılamaz!"** uyarısında bulunur.

```
**********************************************************
                      İŞLEMLER
undo: Son girdiyi listeden çıkarır
redo: Yapılan son undo işlemini geri getirir
yaz : Listeyi eklenme sırasına göre ekrana yazdırır
  q : Programdan çıkış yapar
**********************************************************
Girdi: ali
Girdi: veli
Girdi: 25
Girdi: undo
Girdi: 61
Girdi: yaz

Listl : ali veli 61

Girdi: redo
Undo işlemi yapılmadan redo yapılamaz!
```

Listeye 3 adet veri ekledik
Undo işlemi yaptık
Ekleme işlemi yaptık
Yaz işlemi yaptık
Redo işlemi yaptık
Sonuç

2. Kullanıcı **ekleme – undo – yaz – redo** işlemlerini sırası ile yaptırmak ister ise program ne sonuç verir?

   Birinci durumda anlattığım işlemler bu durumda da geçerli olacaktır. Birinci durum ve bu durum arasındaki fark ise bu sefer yaz işleminden önce undo işleminin olmasıdır. Yaz işleminden önce undo işlemi olduğundan dolayı programımız redo işlemini yapacaktır. Listemizi tekrar yazdırdığımızda redo işleminin yapıldığını göreceğiz.

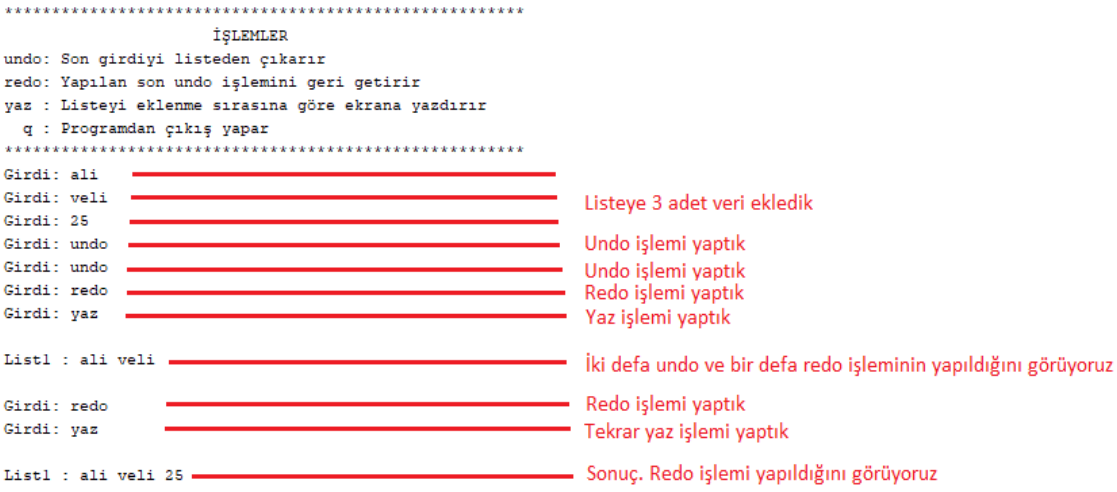
```
**********************************************************
                      İŞLEMLER
undo: Son girdiyi listeden çıkarır
redo: Yapılan son undo işlemini geri getirir
yaz : Listeyi eklenme sırasına göre ekrana yazdırır
  q : Programdan çıkış yapar
**********************************************************
Girdi: ali
Girdi: veli
Girdi: 25
Girdi: undo
Girdi: yaz

Listl : ali veli

Girdi: redo
Girdi: yaz

Listl : ali veli 25
```

Listeye 3 adet veri ekledik
Undo işlemi yaptık
Yaz işlemi yaptık
Undo işleminin yapıldığını görüyoruz
Redo işlemi yaptık
Tekrar yaz işlemi yaparak kontrol ediyoruz
Sonuç. Redo işlemi yapıldığını görüyoruz

3. Kullanıcı **undo – undo – redo – yaz – redo** işlemlerini sırası ile yaptırmak ister ise program ne sonuç verir?

   Birinci durumda anlattığım işlemler bu durumda da geçerli olacaktır. Birinci durum ve bu durum arasındaki fark ise bu sefer yaz işleminden önce redo işleminin olmasıdır.

Yaz işleminden önce redo işlemi olduğu için öncesinde yeterli undo olması şartı sağlandığında redo işlemini yapacaktır. Örnek sıralamada bu yüzden iki defa undo işleminden sonra redo - yaz - redo işlemlerini yaptırdım.



4. Kullanıcı arka arkaya en fazla 5 kez redo işlemi yapabilir.

Yaz İşlemi

Program ilk çalıştığında kullanıcı yaz girer ise liste boş olduğu için ekrana **"Liste boş. Yazdırılacak veri yok!"** uyarısında bulunur. Liste dolu iken kullanıcı yaz girer ise program verileri listeye ekleniş sırası ile ekrana yazar.

Q İşlemi

Kullanıcıdan alınan girdi **q** ise program sonlandırılır.

Sonuç : Kullanıcıdan sonsuz sayıda girdi alındığı için kullanıcının programdan çıkmasını kolaylaştırmak için ekstra olarak **q** işlemini kullandık. Programda belirli bir türde girdi almamızı belirtmediğiniz için string türünü kullandık. Bu sayede int, float türünde veriler girilse bile string şekilde listeye eklenmesini sağladık. Programın çalışma sırasında oluşabilecek ihtimaller denenmiştir ve programda çalışmayan kısma rastlanmamıştır. Programda **undo**, **redo**, **yaz** ve **q** işlemleri düzgün bir şekilde çalışmaktadır.