**YZM 2124**

VERİ YAPILARI

**2022-2023 BAHAR DÖNEMİ**

**1.ÖDEV RAPORU**

Teslim Tarihi: 13.05.2023

Öğrenci Adı: Ramazan YILDIRIM

Öğrenci Numarası:

1. **Giriş**

Bu ödevin amacı N kişilik bir kuyruk oluşturup bu kişilere numara ve random işlem süresi vererek kuyrukta bekleme ve işlem sürelerinin toplamı olan süreleri yazdırıp ortalama süreye göre bir karşılaştırma yapmaktır. Fakat bunu yaparken stack veri yapısından tekrar stack oluşturup daha sonrasında queue veri yapısına aktarılmaktadır. Bu konuda dikkat edilmesi gereken ve ödev dökümanında da belirtilen farklardan bahsedilmesi gerekir.

* 1. ve 1.2 arasındaki farklar :
* 1.1'de list veri yapısı kullanılarak FIFO (First-In-First-Out) kuyruk oluşturulurken, 1.2'de array veri yapısı ve öncelik kuyruğu kullanılarak öncelikli kuyruk oluşturulmaktadır.
* 1.1'de kişilerin kuyruğa geliş sırasına göre işlemleri tamamlanırken, 1.2'de kişilerin işlem süreleri önceliklendirilerek en kısa süreli işlem öncelikli olarak tamamlanmaktadır.
* 1.2'de öncelikli kuyruk kullanıldığı için kişilerin FIFO kuyruğa göre daha az beklediği durumlar olabilir. Bu durumları, işlem süreleri ve sıra numaralarıyla birlikte listeleme imkanı vardır.

1. **Geliştirme ve Çalıştırma Ortamı**

Bu ödev, MacOS işletim sisteminin 13.1 Ventura sürümünde, Visual Studio Code IDE’si ile geliştirilmiştir.

1. **Veri Yapıları, Değişkenler ve Metotlar**

**struct Musteri**: Kod karmaşıklığını önlemek amacıyla içerisinde müşteri numarası (musteriNo) ve işlem süresini (islemSuresi) tutan yapıdır.

**struct MusteriIslem**: Oluşturulan ilk yığındır.

**struct MusteriIslem\_Sirali**: Oluşturulan ikinci yığındır. Oluşturulma sebebi stack veri yapısının özelliğinden dolayı ilk yığına ters giren elemanlar (LİFO) bu ikinci yığına tekrardan sıralı bir şekilde eklenir.

**void push(int \_musteriNo)**: Parametresi sayesinde main()’ de elemanları ilk yığına ters bir şekilde ekler.

**void Musteri\_StacktoStack( )**: push( ) metodu sayesinde ilk yığına ters bir şekilde eklenen elemanları ikinci yığına sıralı bir şekilde atar.

**void MusteriIslemListele( )**: İlk yığındaki elemanların ters bir şekilde alındığını ekranda gösterir.

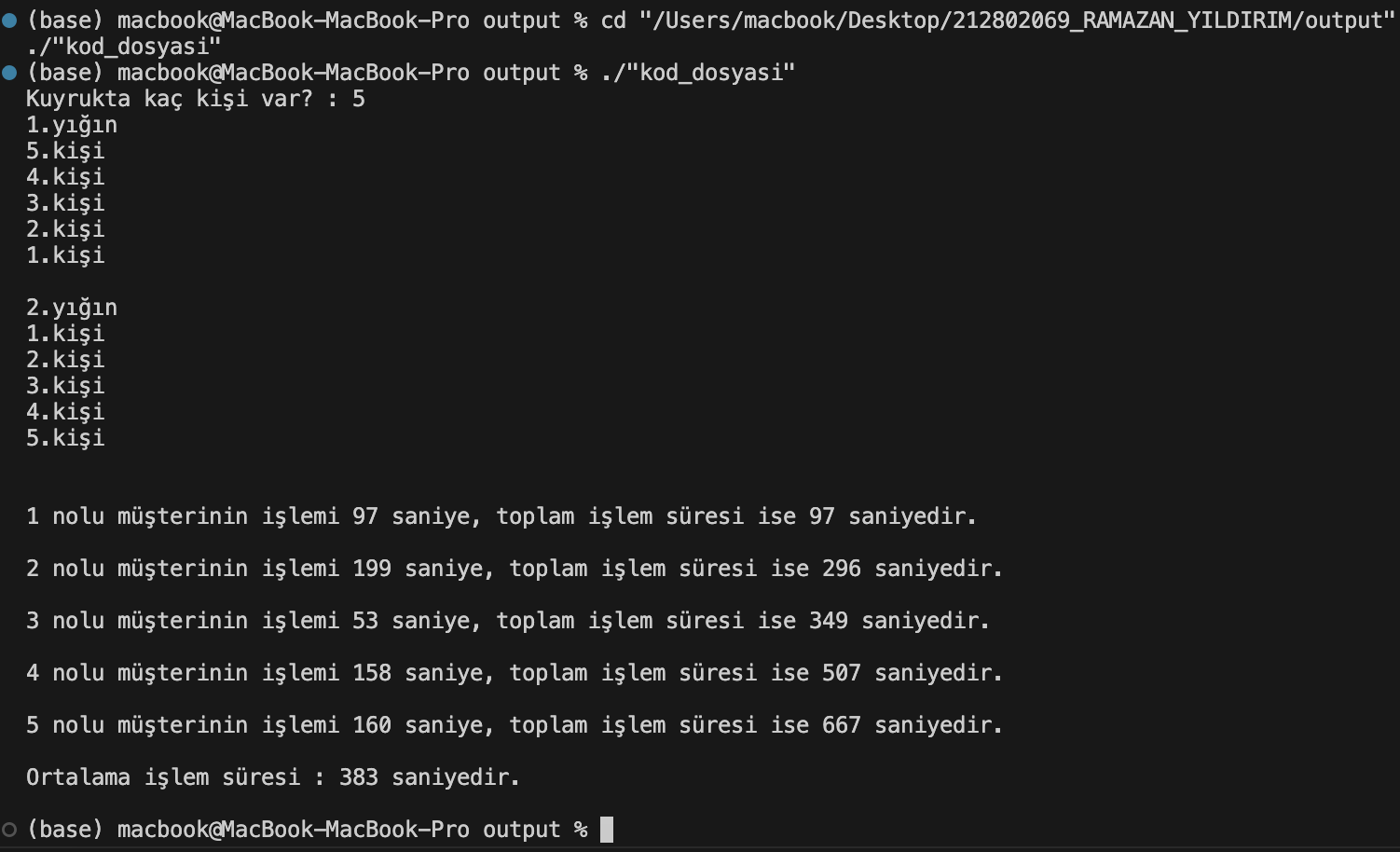
**void MusteriIslemListeleSirali( )**: İlk yığındaki ters elemanların artık ikinci yığında sıralı bir şekilde olduğunu ekranda gösterir.

**void CreateQueue( )**: Elemanları sıralı tutan ikinci stackten bir kuyruk oluşturur ve bu elemanları, elemanların kuyrukta kalma sürelerini ve toplam işlem sürelerini ekrana yazdırır. Son olarak yazdırdığı bu verilere göre bir kişinin ortalama işlem süresini yazdırır.

1. **Program Akışı**

* Kullanıcıdan kuyruk için kişi sayısı istenir.
* Bir döngü sayesinde push( ) metodu aracılığıyla alınan kişi sayısı da baz alınarak stacke elemanlar eklenir.
* Sırasıyla yukarıda her birinin görevleri tek tek belirtilen Musteri\_StacktoStack( ), MusteriIslemListele( ), MusteriIslemListeleSirali( ), CreateQueue( ) metotları çağrılır ve program sonlanır.

1. **Sonuç**

****

Elde edilen kazanımları belirtmeden önce hazırlanan programın bir örnek üzerinden çalıştırılıp bu örneğe göre değerlendirilmesinde fayda var. Bunun için kuyrukta 5 kişinin olduğunu varsayalım. Bu 5 kişiye kuyruk veri yapısında rastgele atanan bu işlem süreleri içinde geliştirilen işletim sistemine göre değişmekle beraber içinde geliştirmiş olduğum macOS işletim sisteminde programı her çalıştırdığımda rastgele olarak hep aynı sayılar getirildi ve bu değerlerin işletim sistemine özgü olduğunu öğrenmiş oldum.

Örneğe tekrar dönecek olursak 5 kişilik bu kuyruğun ortalama işlem süresi de dahil olmak üzere - bunu da eklemenin daha iyi olacağını düşünüyorum. – FİFO kuyruğuna göre küçükten büyüğe sıralamasını yapacak olursak:

( 1. Müşteri : 97 saniye, 2. Müşteri : 296 saniye, 3. Müşteri : 349 saniye,

Ortalama : 349 saniye, 4. Müşteri : 507 saniye, 5. Müşteri : 667 saniye ) gibi bir sıralama yaparız. Yani 4. Ve 5. müşteriler daha fazla bekliyorlar.