**VERİ YAPILARI YAZ**
**OKULU 2.ÖDEV**

**İSİM:** UMUT

**SOYİSİM:** DANIŞ

**NUMARA:** G191210097

**İSİM:** MAZHAR FURKAN

**SOYİSİM:** ÇELEBİ

**NUMARA:** G191210033

## 1. BİZDEN İSTENEN NEDİR?

Program çalıştığı gibi, 0-256 arası sayılarından oluşan ve sayıların arasını birer boşluğun ayırdığı Sayilar.txt dosyası okunacak. Her satır okunurken Yığıta sayılar eklenecektir. Dosyadan yeni okunan sayı, çift ve o an işlem yapılan Yığıttan çıkmak üzere olan sayıdan büyük ise yeni bir Yığıta eklenecektir. Aksi halde var olan Yığıta eklenmeye devam edecektir. Bu şekilde dosyadaki bir satır okunduktan sonra her Yığıt boşaltılıp ayrı bir ikili arama ağacına eklenecektir. Ağaçta aynı değer varsa o değer ikili arama ağacına eklenmeyecektir. Bir satırda bulunan yığıt sayısı kadar ikili arama ağacı oluşturulmalıdır. Her satır için, oluşan ikili arama ağaçlarından en büyük yüksekliğe sahip ikili arama ağacı seçilecektir. Eğer yükseklikler eşit olursa düğüm değerleri toplamı büyük olan ağaç seçilecektir. Toplam değerleri de eşit olan ağaçlardan önce oluşturulan ağaç seçilecektir. Seçilen maksimum yüksekliğe sahip ikili arama ağacı postorder okunup sayısal değerlerin ASCII karakter karşılıkları ekrana yazılıp ekranda 10 milisaniye beklenip (sleep fonksiyonu), bir sonraki satır için yukarıdaki aynı işlemler uygulanıp tekrar karakterler ekrana yazılacaktır. Dosya okuma bittiği zaman program da sonlanacaktır.

## PROGRAMI NASIL KODLADIM ve PROGRAM NASIL ÇALIŞIR

DugumAgac, Yigin ve BinaryBulma sınıflarını tanımladım. DugumAgac düğümleri ağacın elemanlarını temsil ediyor, Yigit yığıt yapısı kullanılarak veriler saklanıyor ve BinaryBulma ise ağaç işlevlerini sağlıyor. DugumAgac sınıfı, ağaç düğümlerini temsil ediyor ve her düğümün verisi ve sol-sağ alt düğüm işaretçilerini bulunduruyor.Stack sınıfı, verilerin yığıt yapısı kullanılarak saklandığı veri yapısını temsil ediyor. Veri ekleme, çıkarma, en üstteki elemana erişim ve boş olup olmadığını kontrol işlemlerini burada yaptım. BinaryBulma sınıfı, ikili arama ağacı veri yapısının işlevselliğini sağlıyor. Ağaca eleman ekleme, yükseklik hesaplama, toplam hesaplama, temizleme ve postorder gezisi gibi işlevler burada yer alıyor. Main fonksiyonuna girince ilk önce "Sayilar.txt" dosyası açıyorum. Dosya açılamazsa hata mesajı verdirdim ve program sonlandırdım. Yigin ve BinaryBulma nesneleri oluşturdum. Bu nesneleri, en büyük yüksekliği ve toplamı içeren ağacı tutmak için kullandım. Metin dosyası satır satır okudum. Her satırı, std::istringstream kullanılarak parçaladım ve her bir sayıyı işledim.Sayı çiftse ve yığıt boş değilse, yığıttaki sayılar kullanılarak yeni bir ağaç oluşturdum. Oluşturulan yeni ağacın yüksekliği ve toplamı, daha önce belirlenen en büyük ağacın yüksekliği ve toplamını geçerse, en büyük ağacı güncelledim. Çift sayılar yığıta ekledim. Dosya satırı sonuna gelindiğinde, yığıttaki sayılar kullanılarak bir ağaç oluşturdum ve yükseklik ve toplam kriterleri kontrol ettim. Yeni oluşturulan ağacın yüksekliği ve toplamı, en büyük ağacın yüksekliği ve toplamını geçerse, en büyük ağacı güncelledim. En büyük yükseklik ve toplama sahip ağacın düğümleri postorder gezilerek biriktirdim ve sonuçları ASCII tablosuna göre yazdırdım. Son olarak en büyük ağacın içeriğini temizledim.

## 2. KULLANDIĞIM SINIFLAR VE METODLAR

Class BinaryBulma

Class Yigin

Class DugumAgac

void ekle Ağaca yeni bir düğüm eklemek için.

void temizle Ağacı temizler

void geziciPost(DugumAgac* dugumKonum, std::string& output) Postorder gezisi yaparak düğüm değerlerini biriktirir

void temizle(DugumAgac* dugumKonum) Düğümden başlayarak ağacı temizler (rekürsif)

void push(int val) Yığına yeni bir eleman eklemek için