1. **Giriş:**

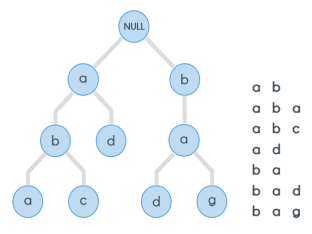
Bu ödevde, Karakter Ağacı ile bir Giriş Sistemi tasarlamanız bekleniyor. Bu denemenin birincil amacı, ağaç veri yapısı üzerinde çalışmanızı sağlamaktır.

1. **Birkaç Bilgilendirme:**

Dizeler temel olarak çeşitli programlama problemleri için en önemli ve yaygın konular olarak görülebilir. Dizi işleme, gerçek dünyadaki çeşitli uygulamalara da sahiptir:

* Arama motorları
* Genom Analizi
* Veri analizi

Metin biçiminde bize sunulan tüm içerik, sadece dizgilerden başka bir şey olarak görülebilir. Bir Karakter ağacı, bir dizenin önekine dayanan özel bir ağaç veri yapısı biçimidir. Bir dizginin öneki herhangi bir n harfinden başka bir şey değildir, n <= | S | Bu, bir dizgenin başlangıcından itibaren başlayarak düşünülebilir. Bir karakter ağacı, bir grafik gibi görselleştirilebilen dizeleri saklamak için kullanılan özel bir veri yapısıdır. Bu düğümler ve kenarlardan oluşur. Her düğüm en fazla 26 çocuktan oluşur ve kenarlar her ebeveyn düğümünü çocuklarına bağlar. Bu 26 işaretçi, İngiliz alfabesinin 26 harfinin her biri için işaretçilerdir. Her düğüm için ayrı bir kenar korunur. Dizeler, bir karakter ağacındaki öneklerine göre yukarıdan aşağıya doğru saklanır. Uzunluk 1'in tüm önekleri 1. seviyeye kadar saklanır, uzunluk 2'nin tüm önekleri 2. seviyeye kadar ve daha sonra sıralanır. Örneğin, aşağıdaki şemaya dikkat edin:



Şimdi, neden tek bir dizgeyi işlemek için bir karakter ağacı gibi bir veri yapısını kullanmalıyız? Aslında, Karakter ağaçları genellikle tek bir dizgeden ziyade dizgiler gruplarında kullanılır. Birden çok dizge verildiğinde, onlara dayalı çeşitli sorunları çözebiliriz. Örneğin, bir İngilizce sözlüğü ve tek bir dizgiyi göz önünde bulundurun, s dizesiyle eşleşen sözlük dizelerinden azami uzunluğun önekini bulun. Bu problemi saf bir yaklaşımla çözmek, verilen dizgenin önekini, sözlükteki diğer her kelimenin ön ekiyle eşleştirmemizi ve maksimum değeri not etmemizi gerektirir. Bu, alacağı süreyi göz önünde bulundurarak pahalı bir süreçtir. Karakter ağaçları bu problemi çok daha verimli bir şekilde çözebilir. En uzun önekin uzunluğunu aramamız gereken türden her Sorguyu işlemeden önce, önce tüm kelimeleri sözlüğe eklememiz gerekir. Bir Karakter ağacı, kök düğümü olarak adlandırılan özel bir düğümden oluşur. Bu düğümün gelen kenarları yoktur. Sadece alfabedeki her harf için en fazla 26 giden kenar içerir ve karakter ağacının köküdür. Bu nedenle, herhangi bir dizenin bir karakter ağacına eklenmesi, kök düğümden başlar. Birinci uzunluğun tüm önekleri, kök düğümün doğrudan çocuklarıdır. Ek olarak, uzunluk 2'nin tüm önekleri birinci seviyede bulunan düğümlerin çocukları haline gelir.

1. **Deney:**

Bu deneyde, Karakter Ağacı ile bir Giriş Sistemi oluşturan bir uygulama yazmanız beklenmektedir. Uygulama, geçerli dizindeki input.txt dosyasını alır ve içeriğini okur. Karakter ağacını oluşturmak için, ekleme, silme ve yenileme için bazı komutlar vardır. Komutların Yapısı:

**-a kullanıcı adı şifre** # kullanıcı adı verilen parola ile ağaca ekle

**-s username** # verilen kullanıcı adı ile arama yap ve şifreyi gir

**-q username** **password** # verilen kullanıcı adı ile şifre için sorguyu gönder

**-d kullanıcı adı** # kullanıcı adı ve şifresini sil.

**-l** # ağacı listele

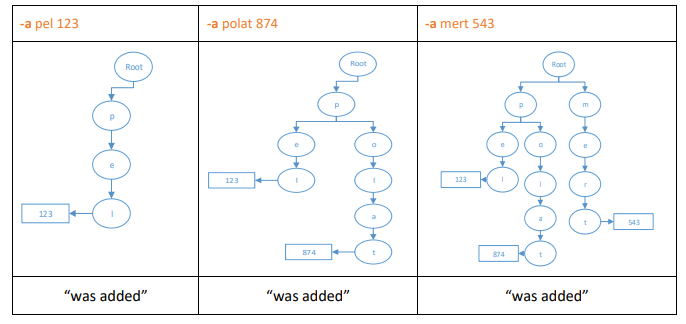
***-a komutuyla;*** Uygulama, verilen kullanıcı adı karakterini karakter olarak okuyacak ve ağaca ekleyecektir. Kullanıcı adının son karakteri ile şifreyi saklayın.

\* Kullanıcı adının ilk karakteri kök düğüm tarafından referans alınmazsa, kullanıcı adı kök düğümden başlayarak ağaca eklenir.

\* Kullanıcı adının ilk n karakteri ağaçta mevcutsa, son karakterler için nn düğümünde bir dal oluşur.

\* Kullanıcı adının son karakteri olan düğümün verilen kullanıcı adı için şifreyi tutması gerekir.

\* Belirtilen kullanıcı adıyla aynı kullanıcı adı varsa, uygulama “ayrılmış kullanıcı adı” yazan bir çıktı verecektir.



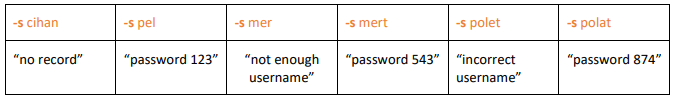
***–s komutuyla;*** Uygulama verilen kullanıcı adını okuyacak ve sorgulanan kullanıcı adına göre bir bilgi verecektir.

\* Eğer kullanıcı adının ilk karakteri kök düğüm tarafından referans alınmazsa, uygulama “kayıtsız” bir çıktı verecektir.

\* Kullanıcı adının ilk n karakteri ağaçta varsa, ancak kalan değilse, uygulama "yanlış kullanıcı adı" olan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta bulunuyorsa, ancak son karakterin şifresi yoksa, uygulama "yeterli kullanıcı adı" olmayan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta bulunuyorsa ve son karakterin şifresi varsa, uygulama "xxx şifresini" veren bir çıktı verecektir.



***–q komutuyla;*** Uygulama kullanıcı adı ve şifresini okuyacak. Eşleşmeye göre, bir bilgi döndürecektir.

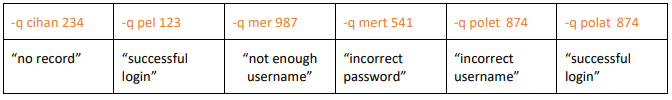
\* Eğer kullanıcı adının ilk karakteri kök düğüm tarafından referans alınmazsa, uygulama “kayıtsız” bir çıktı verecektir.

\* Kullanıcı adının ilk n karakteri ağaçta varsa, ancak kalan değilse, uygulama "yanlış kullanıcı adı" olan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta bulunuyorsa, ancak son karakterin şifresi yoksa, uygulama "yeterli kullanıcı adı" olmayan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta mevcutsa, ancak son karakter verilenden farklı bir parolaya sahipse, uygulama "yanlış parola" olan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta mevcutsa ve son karakter verilen parola ile aynıysa, uygulama "başarılı giriş" diyen bir çıktı verecektir.



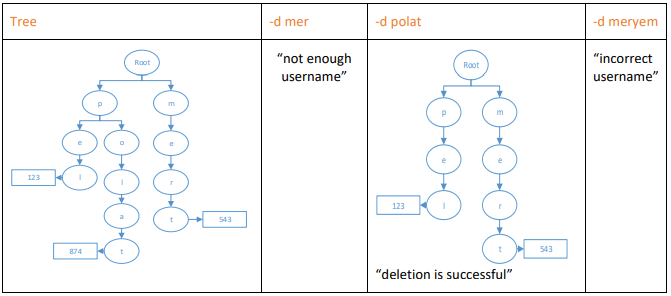
***–d komutuyla;*** Verilen kullanıcı adı ile uygulama kullanıcı adını ve şifresini ağaçtan silecektir.

\* Eğer kullanıcı adının ilk karakteri kök düğüm tarafından referans alınmazsa, uygulama “kayıtsız” bir çıktı verecektir.

\* Kullanıcı adının ilk n karakteri ağaçta varsa, ancak kalan değilse, uygulama "yanlış kullanıcı adı" olan bir çıktı verecektir.

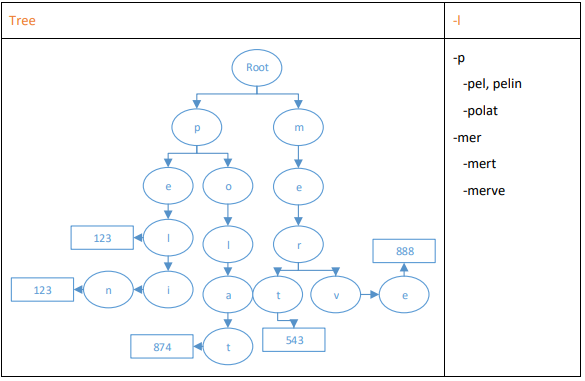
\* Kullanıcı adının tüm karakterleri ağaçta bulunuyorsa, ancak son karakterin şifresi yoksa, uygulama "yeterli kullanıcı adı" olmayan bir çıktı verecektir.

\* Kullanıcı adının tüm karakterleri ağaçta bulunuyorsa ve son karakterin şifresi varsa, uygulama başka bir kullanıcı adına bağlı olmayan tüm düğümleri siler. Ardından “silme işleminin başarılı” olduğu sonucunu verecektir.



***–l komutuyla;*** Uygulama, tüm kullanıcı adlarını ön sipariş geçişine göre listeleyecektir.

\* Ağacın ilk seviyesinden sonra, her dal yeni bir sekmeyle görüntülenecektir.



Önemli konular:

* Uygun bir Makefile olmayan projeler derecelendirilmez.
* Ağacınızdaki tüm terimler küçük harf olmalıdır.
* Programınızı göndermeden önce “dev.cs.hacettepe.edu.tr” adresinde test edin. Gönderiniz sadece bu makinede ve bu makinede derlenecek ve yürütülecektir.
* Rapor 30 puan olacak.
* Lütfen tasarımınızı, veri yapılarınızı ve algoritmalarınızı açıklayın.
* Gereksiz karmaşa olmadan gerekli detayları verin.
* Rapor: <ftp://ftp.cs.hacettepe.edu.tr/pub/dersler/genel/FormatForLabReports.doc>