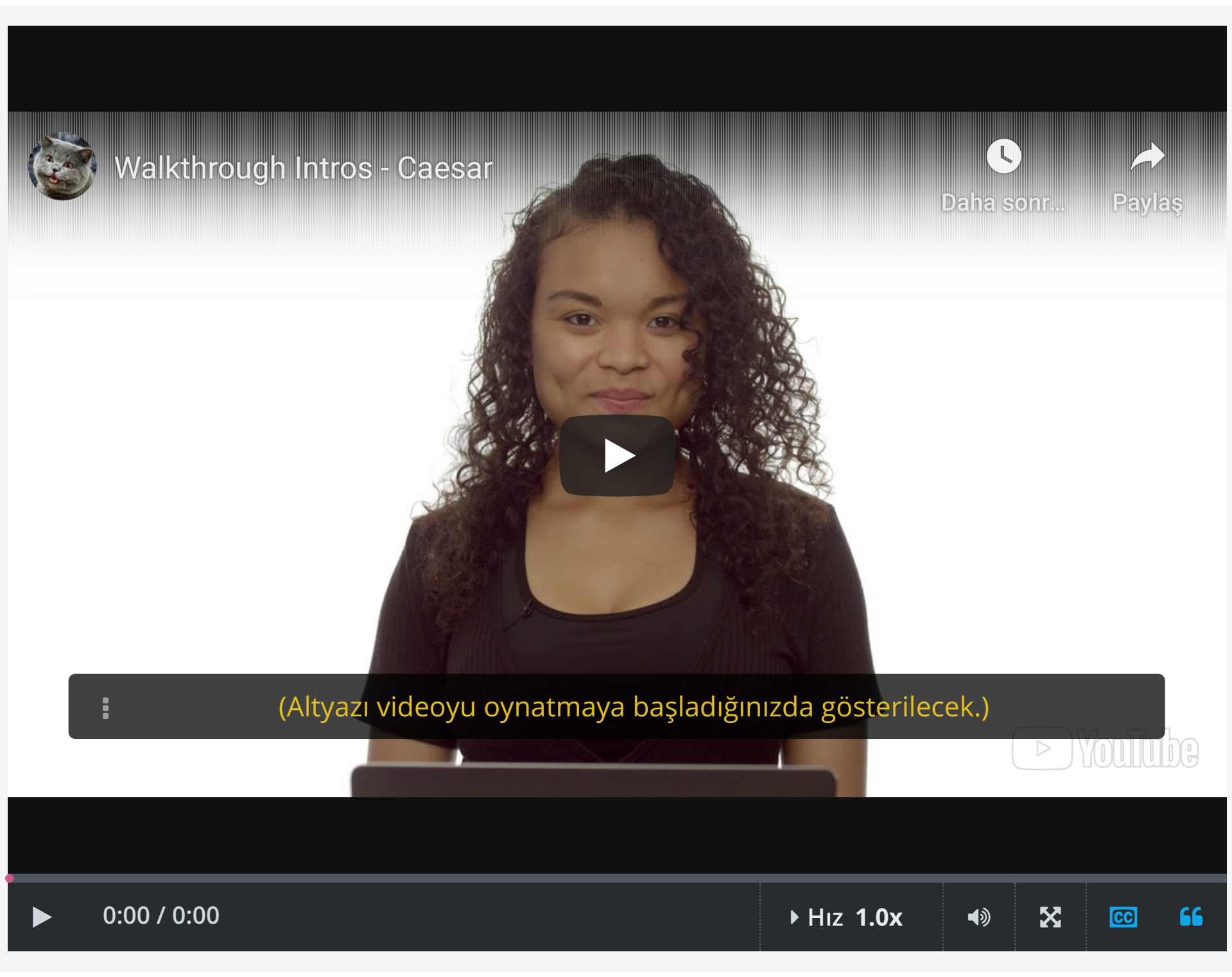
"Caesar" Görevi Tanımı

Bu sayfaya yer imi koy

Video



Altyazının başlangıcı. Sona atla.

Diyelim ki bir arkadaşınıza
veya sizin için özel birine
gizli bir not göndereceksiniz.
Ama kendisi sınıfın öbür ucunda.
Notu yazıp kendisine ulaştırmaya başlıyorsunuz.
O zaman, notu alan herkes içine ne yazdığınızı görecek!
Ama biz bunu istemiyoruz.
Belki de, arkadaşımızla aramızda gizli bir kod oluşturabiliriz.
Mesela her harfi birkaç sıra oynayabiliriz.

Öncelikle CS50 Sandbox'ta bu sayfayı açın ve aşağıdakileri takip edin:

Et tu? (Latince, Sen de mi?)

Sözde Sezar (evet, bildiğiniz Sezar), içlerindeki her harfi bir dizi yere kaydırarak gizli mesajları “şifreliyordu” (yani geri dönüşümlü bir şekilde saklıyordu). Örneğin, Sezar A'yı B, B olarak C, C olarak D,... ve alfabetik olarak Z'yi A olarak yazabilir ve böylece birine HELLO demek için IFMMP yazabilir. Sezar'dan bu tür mesajlar aldıktan sonra, alıcıların harfleri ters yönde aynı sayıda yere kaydırarak “şifresini çözmesi” gerekecektir.

Bu “kriptosistem” in gizliliği sadece Sezar'a ve bir sırrı bilen alıcılara, Sezar'ın harfleri ne kadar değiştirdiğine dayanıyordu (örneğin 1 değiştirme). Modern standartlara göre pek de güvenli değil. Ama, hey, belki de dünyada bunu yapan ilk kişi iseniz, oldukça güvenli!

Şifrelenmemiş metin genellikle düz metin olarak adlandırılır. Şifrelenmiş metne genellikle şifre metni denir. Ve şifreli metni oluşturmak için kullanılan sırra ise anahtar denir.

Net olmak gerekirse, HELLO'yu 1 anahtarıyla şifreleyerek (1 kez kaydırarak) IFMMP'yi işte böyle elde ediyoruz:

düzyazı	H	E	L	L	O
+ anahtar	1	1	1	1	1
= şifremetni	I	F	M	M	P

Daha resmi olarak, Sezar’ın algoritması (yani şifre), her harfi *k* konumlarına "döndürerek" mesajları şifreler. Daha resmi olarak, *p* bir düz metinse (yani şifrelenmemiş bir mesajsa), *p(i)* *p*'deki *İ'inci* karakterdir ve *k* gizli bir anahtardır (yani, negatif olmayan bir tamsayıdır), ardından şifreleme metnindeki her bir *c(i)* şu şekilde hesaplanır:

c(i) = (p(i) + k) % 26

burada % 26 ile gösterilen yer "26'ya bölündüğünde kalan" anlamına gelir. Bu formül belki de şifreyi olduğundan daha karmaşık hale getirir, ancak algoritmayı tam olarak ifade etmenin sadece kısa bir yoludur. Gerçekten de, sırf bu konuları tartışmak için A (veya a) 'yı 0, B (veya b)' yi 1,..., H (veya h) 7, I (veya i) 8,... ve Z (veya z) 25 olarak da düşünebilirsiniz. Diyelim ki Sezar bu sefer, k anahtarı olarak 3’ü kullanıp gizlice “Hi” demek istiyor. Ve böylece onun düz metni *p*, yani Hi. Bu durumda düz metnin ilk karakteri p(0), yani H (yani 7) ve düz metnin ikinci karakteri ise p(1), yani i’dir (yani 8). Şifreleme metninin ilk karakteri olan c(0), K'dir ve şifreleme metninin ikinci karakteri olan c(1), L'dir. Nedenini görebiliyor musunuz?

Ödeviniz olarak, Sezar'ın şifresini kullanarak mesajları şifrelemenizi sağlayan Sezar adlı bir program yazmanızı istiyoruz. Kullanıcı programı çalıştırırken, komut satırında argüman vererek gizli mesajda anahtarın ne olması gerektiğine karar vermelidir. Kullanıcının anahtarının bir sayı olacağını varsaymamalıyız; yine de, eğer bir sayı ise, pozitif bir tamsayı olacağını varsayabilirsiniz. İşte programın nasıl çalışabileceğine dair birkaç örnek. Örneğin, kullanıcı 1 anahtarını ve HELLO düz metnini girerse:

```
$ ./caesar 1
plaintext:  HELLO
ciphertext: IFMMP
```

Kullanıcı 13 anahtarını ve "hello, world" düz metnini sağlarsa program şu şekilde çalışabilir:

```
$ ./caesar 13
plaintext:  hello, world
ciphertext: uryyb, jbeyq
```

Ne virgül ne de boşluğun şifre tarafından “kaydırılmadığına” dikkat edin. Sadece alfabetik karakterleri kaydırın!

Bir tane örneğe daha ne dersiniz? Kullanıcı tekrar 13 anahtarını daha karmaşık bir düz metinle sağlarsa program şu şekilde çalışabilir:

```
$ ./caesar 13
plaintext:  be sure to drink your Ovaltine
ciphertext: or fher gb qevax lbhe Binygvar
```

Orijinal mesajın korunduğuna dikkat edin. Küçük harfler küçük, büyük harfler büyük kalır.

Peki ya bir kullanıcı bize yardımcı olmazsa?

```
$ ./caesar HELLO
Usage: ./caesar key
```

Cidden olmazsa?

```
$ ./caesar
Usage: ./caesar key
```

Ya da hatta...

```
$ ./caesar 1 2 3
Usage: ./caesar key
```

Siz deneyin:

Bu probleme ekibin bulduğu çözümü denemek için bu sandbox'ta aşağıdaki komutu çalıştırın (tabii ki *key* yerine geçerli bir tamsayı yazarak):

```
./caesar key
```

Peki, bu probleme nasıl başlamalıyız? Bu soruna adım adım yaklaşalım:

Sözde Kod (Pseudocode)

Öncelikle, bu programı uygulayan kodun nasıl yazıldığından emin olmasanız bile (şimdilik!), Ceaser CS50 Lab'de yer alan pseudocode.txt dosyasına bu programı uygulayan bir sözde kod yazın. Sözde kod yazmanın tek bir doğru yolu yoktur, yalnızca kısa cümleler yeterlidir. Mike Smith'i bulmak için sözde kodu nasıl yazdığımızı hatırlayın. Olasılıkla, sözde kodunuz bir veya daha fazla işlev, koşul, Bool ifadesi, döngü ve / veya değişken kullanacaktır (ya da ima edecektir!)

Spoiler:

Bunu yapmanın birden fazla yolu var, işte sadece bir tanesi!

1. Programın bir komut satırı değişkeniyle çalıştırılıp çalıştırılmadığını kontrol edin.
2. Tüm karakterlerin rakam olduğundan emin olmak için sağlanan argümanı yineleyin.
3. Bu komut satırı argümanını string’den int'e dönüştürün.
4. Kullanıcıdan düz metin isteyin.
5. Düz metnin her karakteri üzerinde yineleme yapın:

   i. Büyük harfse, şifreye göre kaydırın, harf büyüklüğünü koruyun, sonra kaydırılmış karakteri yazdırın.
   ii. Küçük harfse, şifreye göre kaydırın, harf büyüklüğünü koruyun, sonra kaydırılmış karakteri yazdırın.
   iii. İkisi de değilse karakter neyse onu yazdırın.

6. Yeni satır yazdırın.

Bu sözde kodu gördükten sonra kendinizinkini buna bakarak düzenleyebilirsiniz, ancak bizimkini alıp direkt kopyalayıp yapıştırmayın!

Komut Satırı Argümanlarını Sayma

Sözde kodunuz ne olursa olsun, ek işlevsellik eklemeden önce yalnızca programın tek bir komut satırı bağımsız değişkeniyle çalıştırılıp çalıştırılmadığını kontrol eden C kodunu yazalım.

Özellikle, CS50 Lab'de "caesar.c" yi şu şekilde değiştirin: kullanıcı tam olarak bir komut satırı argümanı sağlarsa, "Success" yazar; kullanıcı herhangi bir komut satırı argümanı veya iki ya da daha fazlasını sağlamazsa, "Usage: ./caesar key" yazdırır ve main hemen 1 değerini (bir hata olduğunu gösterir) döndürür. Unutmayın, bu anahtar doğrudan komut satırından geldiğinden ve "get_string" yoluyla gelmediğinden, kullanıcıdan yeniden girdi isteme fırsatımız yoktur. Ortaya çıkan programın davranışı aşağıdaki gibi olmalıdır.

```
$ ./caesar 20
Success
```

ya da

```
$ ./caesar
Usage: ./caesar key
```

ya da

```
$ ./caesar 1 2 3
Usage: ./caesar key
```

İpuçları

- Programınızı make ile derleyebileceğinizi hatırlayın.
- Printf ile yazdırabileceğinizi hatırlayın.
- Argc ve argv'nin komut satırına girilenler hakkında bilgi verdiğini hatırlayın.
- Programın adının (burada, ./caesar) argv[0]'da olduğunu hatırlayın.

Anahtar Erişimi

Artık programınız (umuyoruz ki!) öngörülen şekilde giriş kabul ettiğine göre, şimdi başka bir adım atmanın zamanı geldi.

Programımızı, teknik olarak bir argüman (anahtar) giren, ama girdiği argüman bir tamsayı olmayan kullanıcılara karşı korumamız gerektiğini hatırlayın, örneğin:

```
$ ./caesar xyz
```

Ancak anahtarı geçerlilik açısından analiz etmeye başlamadan önce, gerçekten okuyabildiğimizden emin olalım. "Caesar.c" dosyasını, kullanıcının yalnızca bir komut satırı argümanı sağlayıp sağlamadığını denetlemekle kalmayıp, bunu doğruladıktan sonra, bu tek komut satırı argümanını da yazdıracak şekilde değiştirin. Örneğin, bu davranış şöyle görünebilir:

```
$ ./caesar 20
Success
20
```

İpuçları:

- Argc ve argv'nin komut satırına girilenler hakkında bilgi verdiğini hatırlayın.
- Argv'nin bir string dizisi olduğunu hatırlayın.
- Printf ile %s’i yer tutucu olarak kullanıp bir string yazdırabileceğimizi hatırlayın.
- Bilgisayar bilimcilerin 0'dan başlayarak saymayı sevdiğini hatırlayın.
- Köşeli parantez kullanarak argv gibi bir dizinin tek tek öğelerine erişebileceğimizi hatırlayın, örneğin: argv [0].

Anahtarı Doğrulama

Artık anahtarı nasıl okuyacağınızı bildiğinize göre, onu analiz edelim. CS50 Lab üzerindeki Caesar.c dosyasını, verilen komut satırı argümanı yazdırmak yerine, o komut satırı argümanının her karakterinin ondalık bir rakam (ör. 0, 1, 2 vb.) olup olmadığını denetleyecek şekilde değiştirin ve bunlardan herhangi biri değilse, Usage: ./caesar key iletisini yazdırdıktan sonra hemen sona ersin. Ancak argüman yalnızca rakamlardan oluşuyorsa, bu string’i (argv’nin, bu string’ler sayı gibi görünse bile bir string dizisi olduğunu hatırlayın) gerçek bir tam sayıya dönüştürmelisiniz. Sonra da bu tam sayıyı priftf ile %i üzerinden yazdırın. Örneğin, bu davranış şöyle görülebilir:

```
$ ./caesar 20
Success
20
```

ya da

```
$ ./caesar 20x
Usage: ./caesar key
```

İpuçları

- Argv'nin bir string dizisi olduğunu hatırlayın.
- Bu arada bir string’in sadece bir karakter(char) dizisi olduğunu hatırlayın.
- String.h başlık dosyasının, string’lerle çalışan bir dizi yararlı fonksiyon içerdiğini hatırlayın.
- Uzunluğunu biliyorsanız, bir string’in her bir karakterini yinelemek için döngü kullanabileceğimizi hatırlayın.
- Ctype.h başlık dosyasının, karakterler hakkında bize bilgi veren bir dizi yararlı fonksiyon içerdiğini hatırlayın.
- Programımızın başarıyla tamamlanmadığını belirtmek için sıfır olmayan değerleri main öğesinden döndürebileceğimizi hatırlayın.
- Printf ile %i’yi yer tutucu olarak kullanıp bir tam sayı yazdırabileceğimizi hatırlayın.
- atoi fonksiyonunun, bir sayı gibi görülen bir string'i o sayıya dönüştürdüğünü hatırlayın.

Kaputun Altına Bir Göz Atmak

İnsanlar olarak, “H + 1 = I” diyebildiğimiz gibi, yukarıda açıklanan formülü sezgisel olarak anlamak bizim için kolaydır. Fakat bir bilgisayar aynı mantığı anlayabilir mi? Hadi öğrenelim.Şimdilik, kullanıcının sağladığı anahtarı geçici olarak göz ardı edeceğiz ve bunun yerine kullanıcıdan gizli bir mesaj isteyecek ve tüm karakterlerini sadece 1 kaydırmaya çalışacağız.

Caesar.c fonksiyonunu anahtarı doğruladıktan sonra kullanıcıdan bir string (“açık metin” istediğimizi belirterek) ister ve sonra tüm karakterlerini 1 kaydırarak “şifre metin” çıktısını verir, sonra da yeni satıra geçeriz. Programınız daha sonra main’den 0 döndürerek çıkmalıdır. Ayrıca bu noktada, önceden “Success” yazdırdığımız kod satırını da silebiliriz. Bütün bunlarla programın davranışı şöyle olur:

```
$ ./caesar 1
plaintext:  hello
ciphertext: ifmmp
```

İpuçları

- Düz metindeki her karakteri yinelemeye çalışın ve kelimenin tam anlamıyla 1 ekleyin, ardından yazdırın.
- Eğer c, C dilinde char türünde bir değişkense, printf ("% c", c + 1) çağrıldığında ne olur?

Şimdi Sıra Sizde

Şimdi her şeyi birbirine bağlama zamanı! Karakterleri 1 kaydırmak yerine, caesar.c dosyasını değiştirerek gerçek anahtar değerine kaydırın. Ve büyük-küçük harf farkını koruduğunuzdan emin olun! Büyük harfler büyük, küçük harfler küçük olmalı ve alfabetik olmayan karakterler değişmemelidir.

İpuçları

- Modulo (yani, geri kalan) operatörünü (%), Z'den A'ya kadar sarmalamak için kullanmak en iyisidir! Peki nasıl?
- Önceki bölümdeki tekniği kullanarak Z veya z'yi 1'e sarmaya çalışırsak işler garipleşir.
- Noktalama işaretlerini bu tekniği kullanarak sarmaya çalışırsak yine işler garipleşir.
- ASCII'nin yazdırılabilir tüm karakterleri rakamlarla eşlediğini hatırlayın.
- A'nın ASCII değerinin 65 iken a'nın ASCII değerinin 97 olduğunu hatırlayın.
- Printf'yi çağırdığınızda hiç çıktı görmüyorsanız, bunun nedeni muhtemelen geçerli ASCII aralığının dışındaki karakterleri yani 0 ile 127 arasındakileri yazdırmamanızdır. Hangi değerleri yazdırdığınızı görmek için önce karakterleri sayı olarak yazdırmayı (%c yerine %i kullanarak) deneyin ve yalnızca geçerli karakterleri yazdırmaya çalıştığınızdan emin olun!

Şimdi sizin ödeviniz:

Ödevinizi, bir sonraki ekranda çıkan kutuya kodunuzu yazarak göndereceksiniz. Ödevinizi bize göndermeden, aşağıdaki ödev tanımını okuduğunuzdan emin olun. Sonra da gönderirken dikkat etmeniz gerekenler şunlar:

- Bu ödevden bir puan alacaksınız ve ilerleyişinize işlenecek.
- Ödevi sistemde bize göndermeden önce, mutlaka CS50 Lab'de test etmenizi öneririz.
- Eğer sonucunuz yanlışsa 0 puan, doğruysa 1 puan alacaksınız. Doğru yapana kadar birkaç kez deneme şansınız var.
- Eğer sonucunuz yanlışsa, aşağıda "See Output" kısmından neler olduğunu inceleyebilirsiniz.
- **Cevabınız doğru olduğunda, mutlaka Gönder tuşuna basıp sonucu bize göndermeyi unutmayın!**
- İnternet bağlantınız ve problemlere bağlı olarak, sonuçların otomatik değerlendirilmesi bazen uzun sürebilir. Bu durumda 5-10 dakika beklemeniz gerekebilir. Eğer çalışmıyorsa sayfanızı yenileyebilirsiniz.