**PROJECT EULER PROBLEM ÇÖZÜMLERİ**

**Problem 1**

sum değişkeni 0 olarak ilklendirilir. 3'ten 1000'e kadar 3'er 3'er artan bir for döngüsü açılır ve sum değişkenine her iterasyonda döngünün indisi eklenir. Sonra 5'ten 1000'e kadar bir for döngüsü açılır ve eğer döngünün indisi 15'e tam bölünmüyorsa sum değişkenine eklenir, çünkü 15'e bölünenler ilk açılan for döngüsünde 3'ün katı olduğu için zaten sum değişkenine eklenmişlerdi. Döngülerden sonra elde edilen sum değeri ekrana bastırılır.

**Problem 2**

x ve y değişkenleri fibonacci sayısının ilk 2 terimini temsil etsin ve 1'e ilklendirilsinler, toplam değişkeni 0'a ilklendirilsin. y 4 milyondan küçük olduğu sürece bir while döngüsü dönsün ve içerisinde aşağıdaki işlemler gerçekleşsin:

y = x + y  
x = y - x

Bu işlemlerin sonucunda y ve x değişkenleri bir soru fibonacci terimlerine kayarlar. Eğer y mod 2 0'a eşitse toplam değişkenine y eklenir. Döngüden çıkıldığında toplam değişkeni yazdırılır.

**Problem 3**

N değeri okunur. i değerine 3 atanır. N sayısı 2’ye bölünebildiği kadar bölünür. Bu aşamada en büyük asal çarpan 2’dir. İlk önce 3 atadığımız i sayısına bölünebildiği kadar bölünür. Eğer bu işlem gerçekleşirse en büyük asal çarpan olarak i, l\_prime isimli değişkende saklanır. Asal sayılar tek sayılar olduğundan (2 dışında) i 2 artırılır ve bu işlemler n 1 olana kadar devam eder. N bir olduğunda l\_prime değişkeninde saklanan sayımız bizim n sayısının en büyük asal çarpanıdır.

**Problem 4**

i değişkeni 999 olarak, max değişkeni 0 olarak ilklendirilir. i 100'den büyük eşit olduğu sürece dönen bir while döngüsü içerisinde j değişkenine i değeri atanır, ve içeride j büyük eşit 1 olduğu sürece dönen bir while döngüsü daha oluşturulur. Sonra product değişkenine i\*j çarpımı atanır ve bu çarpım palindrom bir sayıysa ve product max’tan büyükse max değişkenine product atanır. Döngü bittiğinde max istenilen cevabı depolamıştır.

Çarpımın palindrom olup olmadığı isPalindrome fonksiyonuna product gönderilerek kontrol edilir. product string bir ifadeye dönüştürülür ve bu ifadenin yarısına kadar olan karakterlerin sondan yarısına kadar olan karakterlerle aynı olduğu durumda fonksiyon true döndürür. Karakterler aynı değilse fonksiyon false döndürür.

**Problem 5**

Problemde 1, 2, ..., 20 sayılarının en küçük ortak katı isteniyor. Bir while döngüsü açılır, i 0 ile ilklendirilir, her iterasyonda 1 artar. Döngüde 11 ile 20 arasındaki her sayının i ile mod'u alınıp 0'a eşit olup olmadığı kontrol edilir, çünkü 11 ile 20 arasındaki sayılara tam bölünen her sayı 1 ile 20 arasındaki her sayıya da tam bölünür. Yapılan kontrol sonucu eğer bütün mod işlemleri 0'a eşitse döngü durur, bulunan sayı ekrana yazdırılır.

**Problem 6**

sum\_of\_squares isminde bir değişken 0'a eşitlenir. 1'den 100'e kadar for döngüsü çalıştırılır, döngünün indisinin karesi sum\_of\_squares değişkenine eklenir. square\_of\_sum isminde bir değişken 0'a eşitlenir. 1'den 100'e kadar for döngüsü çalıştırılır, döngünün indisi square\_of\_sum değişkenine eklenir. square\_of\_sum değişkeninin karesi alınır.

square\_of\_sum - sum\_of\_squares farkı ekrana yazdırılır.

**Problem 8**

Sayı bir string değişkeni içerisine atılır, maximum 0 olarak ilklendirilir. 1'den 988'e kadar for döngüsü çalıştırılır, içeride de 0'dan 12'e kadar for döngüsü çalıştırılır, dış ve iç döngünün indisleri toplanarak sayının basamaklarına erişilir ve bir değişkende çarpılır. Çarpım maximum’dan büyükse maximum çarpıma eşitlenir. Döngülerden çıkıldığında elde edilen maximum istenilen cevaptır.

**Problem 9**

int a,b,c,ans;  
for (a=0 ; a<(1000/3) ; a++) {  
 for (b=(a+1) ; b<(1000/2) ; b++) {  
 c = 1000 - a - b;  
 if ((c>b) && (c\*c == b\*b + a\*a)) {  
 ans = a\*b\*c;  
 printf("Cevap : %d\n",ans);  
 }  
 }  
}

**Problem 10**

toplam değişkeni 0 olarak ilklendirilir. 2 milyon elemanlık bir dizi açılır ve bütün elemanlar 1'e eşitlenir. Dizi üzerinde 2'den 2 milyona kadar for döngüsü ile gezilir, iterasyon içerisinde eleman 1 ise toplam değişkenine eklenir ve bir while döngüsü ile bu elemanın tam katları dizide 0'a eşitlenir. Döngü bittikten elde edilen toplam istenilen değerdir.

**Problem 12**

Problemde üçgensel sayılarda 500'den fazla çarpanı olan ilk sayının bulunmasını istiyor.

Bir sayının çarpan sayısı, o sayının asal çarpanlarının üstlerinin 1 ile toplanarak çarpılmasıyla elde edilir.

Örneğin 28 = 22\*71 olduğundan 28'in çarpan sayısı (2+1)\*(1+1) ile elde edilebilir.

Yani eğer bir üçgensel sayıdaki asal çarpanların o sayıda kaç kere bulunduğu tespit edilirse, çarpan sayısı elde edilebilir.

Bunun için önce asal sayılar elde edilmelidir.

Asal sayıları elde edebilmek için generatePrimes isimli bir fonksiyon yazılmıştır. Fonksiyon parametre olarak asal sayıların yerleştirileceği bir dizi alır, bu dizinin ismi primes olsun. Fonksiyonun içerisinde elemanları indislerine eşit olan numbers isimli bir dizi daha oluşturulmuştur. (Örnek: numbers[5] = 5)

i for döngüsünün indisi olmak üzere 2'den 100000'e kadar for döngüsü çalıştırılmıştır. numbers’ın i. elemanı eğer 0'değilse bu eleman primes dizisine eklenir ve numbers dizisinde i. elemanın tam katları 0'a eşitlenir. Burada yapılan işlem asal bir sayı bulunduğunda onu primes dizisine eklemek ve asal sayının tam katlarını sıfırlamaktır. Bu yolla primes dizisi asal sayılarla doldurulur.

Bundan sonra ise number\_of\_factors isimli fonksiyon oluşturulur. Bu fonksiyonun görevi ise parametre olarak gelen sayının (Bunu number değişkeni ile gösterelim.) kaç tane çarpanı olduğunu bulmaktır. factors isimli değişken oluşturulup değeri 1'e eşitlenir, bu fonksiyonun döndüreceği değerdir. primes dizisinin indisi i olmak üzere, number’ın yarısı büyük eşit primes[i] olduğu sürece i artırılır. (Çünkü number’ın yarısından büyük bir asal sayı number’ın çarpanı olamaz.) While döngüsünün içinde number’ın değerini depolayan tmp\_number isimli bir değişken oluşturulur, bu değişken primes[i]’ye tam bölündüğü sürece count isimli bir değişkenin değeri 1 artırılır, bu sayede count değişkeninde number’ın içinde primes[i]'den kaç kere geçtiği elde edilmiş olur. Bu count değeri de 1 ile toplanarak factors değeri ile çarpılır ve factors'e eşitlenir.

Son aşama olarak da bir while döngüsü kurulur. Döngü number\_of\_factors(number) küçük eşit 500 koşulu sağlandığı sürece döner ve number değerine bir sonraki üçgensel sayı n \* (n+1) / 2 formulü ile atanır. Döngü 500'den büyük çarpan sayısına sahip number’ı bulduğunda kırılır, ekrana bulunan number yazdırılır.

**Problem 13**

* Problemde verilen bütün sayılar bir diziye atıldı.
* Başlangıç değeri 0 olan sum isimli bir değişkende dizideki elemanlar toplandı.
* tmp\_sum isimli bir değişkene sum değeri atandı.
* Bir while döngüsü ile tmp\_sum 0'a eşit oluncaya kadar 10'a bölündü ve her dönüşte başlangıç değeri 0 olan number\_of\_digits isimli değişkenin değeri 1 artırıldı.
* number\_of\_digits'te toplamın basamak sayısı elde edildikten sonra sum, 10 üzeri (number\_of\_digits - 10) ile tam sayı bölmesi yapılarak toplamın ilk 10 basamağı elde edildi.
* Elde edilen değer ekrana yazdırıldı.

**Problem 14**

longest\_chain en uzun zincir uzunluğunu ve longest\_chain\_starter bu zincirin hangi sayı ile başladığını belirten değişkenler olsun ve 0 ile ilklendirilsinler. 2'den 1 milyona kadar for döngüsü açılır, döngünün içinde next değişkenine döngünün indisi, count değişkenine 0 atanır. next 1 olmadığı sürece while döngüsü açılır ve next mod 2 0 ise next 2'ye bölünür, değilse 3 ile çarpıp 1 eklenir. count 1 artırılır. While döngüsü bittikten sonra ise count longest\_chain’den büyük ise longest\_chain değişkenine count, longest\_chain\_starter değişkenine döngünün indisi atanır. For döngüsü bittiğinde elde edilen longest\_chain\_starter istenilen sayıdır.

**Problem 15**

unsigned long long int result , i , j , n , sonuc;  
printf("n degerini giriniz : ");  
scanf("%llu",&n);  
sonuc = 1;  
j = 1;  
for (i=(n+1) ; i<=(2\*n) ; i++) {  
 sonuc = sonuc \* i / j;  
 j++;  
}  
printf("%llu",sonuc);

**Problem 16**

2 üzeri 1000 hesaplanarak sayi isimli değişkene aktarılır. toplam değişkeni 0 olarak ilklendirilir. sayi 0 olmadığı sürece dönen bir while döngüsü oluşturulur. toplam değişkenine sayi mod 10 eklenir ve sayi 10'a tam sayı bölmesi ile bölünür. Bu şekilde bütün basamaklar toplam değişkenine eklenmiş olur. Döngüden çıkıldığında toplam değişkeni ekrana yazdırılır.

**Problem 17**

bir\_dokuz = "onetwothreefourfivesixseveneightnine";  
on\_doksan = "twentythirtyfortyfiftysixtyseventyeightyninety";  
on\_ondokuz = "teneleventwelvethirteenfourteenfifteensixteenseventeeneighteennineteen";  
yuz = "hundred";  
ve = "and";  
birbin = "onethousand";

Yukarıdakiler string değişkenler olsun. uzunluk fonksiyonu içerisine aldığı string ifadenin karakter sayısını döndürmek üzere aşağıdaki işlemler yapılır.

bir\_dokuz\_icerenler = (9 + 9\*9 + 100) \* uzunluk(bir\_dokuz) (1)  
on\_doksan\_icerenler = (10 + 9\*10) \* uzunluk(on\_doksan) (2)  
on\_ondokuz\_icerlenler = (1 + 1\*9) \* uzunluk(on\_ondokuz) (3)  
yuz\_icerenler = 900 \* uzunluk(yuz) (4)  
ve\_icerenler = (900-9) \* uzunluk(ve) (5)  
bin = uzunluk(birbin) (6)

1. işlemde, içerisinde "one" geçen sayılar hesaplanıp bir\_dokuz'un uzunluğu ile çarpılarak bir\_dokuz ifadelerinin toplam karakter sayısı elde edilmiştir. Yapılan işlemde 1-9 arasında 9 kere, 10-90 arasında 9\*9 kere, ve "one hundred, one hundred and ninefive" gibi durumların başına gelen de 100 tane "one" vardır.

2. işlemde, içersinde "twenty" geçen sayılar hesaplanıp on\_doksan'ın uzunluğu ile çarpılarak on\_doksan ifadelerinin toplam karakter sayısı elde edilmiştir. Yapılan işlemde 10-100 arasında "twenty, twenty nine" şeklinde geçen 10 tane, 100-1000 arasında her yüzde 10 kere olmak üzere 9\*10 kere "twenty" vardır.

3. işlemde, içerisinde "ten" geçen sayılar hesaplanıp on\_ondokuz'un uzunluğu ile çarpılarak on\_ondokuz ifadelerinin toplam karakter sayısı elde edilmiştir. Yapılan işlemde 10-19 arasında 1 tane, 100-1000 arasında ise her yüzde 1 tane olmak üzere 1\*9 tane on\_ondokuz vardır.

4. işlemde, içerisinde "hundred" geçen sayılar hesaplanıp yuz'un uzunluğu ile çarpılarak yuz ifadelerinin toplam karakter sayısı elde edilmiştir. 900 tane "hundred" ifadesi vardır, çünkü üç basamaklı 900 sayı vardır ve bunların hepsinde "hundred" geçer.

5. işlemde, içerisinde "and" geçen sayılar hesaplanıp ve'nin uzunluğu ile çarpılarak ve ifadelerinin toplam karakter sayısı elde edilmiştir. "one hundred, two hundred" gibi ifadeler dışındaki bütün üç basamaklı sayılarda "and" ifadesi vardır, 900 üç basamaklı sayı, 9 tane "100, 200..." şeklinde sayı olduğundan 891 tane "ve" içeren sayı vardır.

6. işlemde, 1000, yani "onethousand" ifadesinin karakter sayısı hesaplanmıştır.

İşlemler sonucunda elde edilen değişkenlerin toplamı istenen sonuçtur.

**Problem 18**

Verilen üçgen triangle isimli bir diziye aktarıldı.

sum isimli özyinelemeli bir fonksiyon yazıldı, fonksiyonun aldığı parametreler sırasıyla triangle dizisi, fonksiyon çağrıldığında dizide bakılan eleman, bu elemanın satır numarası.

15 tane satır olduğu için fonksiyonun temel durum kontrolü girilen satırın 15 olduğu durum, bu durumda fonksiyon triangle[i] değerini döndürüyor.

Eğer temel durumda değilse o zaman şu şekilde ikiye ayrılıyor:

left = triangle[i] + sum(triangle, i+n, n+1);  
right = triangle[i] + sum(triangle, i+n+1, n+1);

Burada left ağaca sol taraftan gidilirse elde edilen toplamları, right ağaca sağ taraftan gidilirse elde edilen toplamları göstermektedir. i+n bir sonraki satırdaki solda bulunan komşu elemanı, i+n+1 ise bir sonraki satırda sağda bulunan komşu elemanı göstermektedir. n+1 ise bir sonraki satırın numarasıdır.

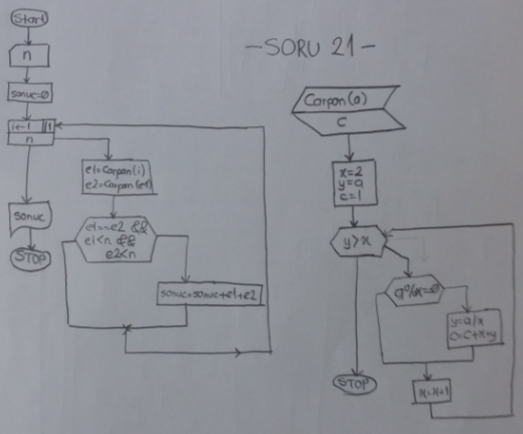
Bundan sonra left ve right değerleri birbirleriyle karşılaştırılır ve büyük olan geri döndürülür. Bu sayede ağacın el altına kadar left ve right ile inilir, yukarı doğru çıkılırken de büyük olan taraf kullanılarak yukarı çıkılır, en tepeye gelindiğinde ise fonksiyon en büyük toplamı geri döndürmüş olur.

Son olarak bu toplam ekrana yazdırılır.

**Problem 20**

factorial\_of\_100 değişkeni 1 ile ilklendirilir. 2'den 100'e kadar bir for döngüsü açılır ve döngünün indisi ile çarpılarak 100 faktöriyel değeri factorial\_of\_100 değişkeninde depolanır. sum değişkeni 0 ile ilklendirilir. Sonra factorial\_of\_100 0 olmadığı sürece bir while döngüsüne girer ve 10'a bölümünden kalan sum değişkenine eklenir ve factorial\_of\_100 10 ile bölünür. While döngüsünden çıkıldığında elde edilen sum istenen sonuçtur.

**Problem 21**



**Problem 22**

İsimler bir diziye atılır. Dizi sıralanır. Toplam skoru depolayacak toplam isminde bir değişken 0 olarak ilklendirilir. For döngüsü ile dizinin her isminin üstünde dolaşılır, bu döngünün içinde bir for döngüsü daha açılarak isimlerinin her karakterinin üzerinde dolaşılır. İç döngüde ilk değeri 0 olan score değişkenine  
 “karakterin ascii değeri - 'A' harfinin ascii değeri + 1"  
 eklenerek ismin harf değerleri toplanır, dizinin indisi ile çarpılarak bulunan sonuç toplam değişkenine eklenir. Döngülerden çıkıldığında elde edilen sonuç istenilen değerdir.

**Problem 24**

Problemde 0, 1..., 9'a kadar olan rakamlardan oluşan, küçükten büyüğe doğru 1 milyonuncu sayı isteniyor. Bu rakamlar sayılar isimli bir diziye atılmış olsun. Bir de istenen sonucu saklayacağımız sonuç isimli 10 gözden oluşan bir dizi olsun.

i değişkeni kalan elemanların i! dizilişte olduğunu belirtsin.

Başı 0 olan bir dizinin geriye kalan elemanlarının diziliş sayısı 9!'dir. i’nin ilk değeri 9 olsun.

toplam değişkeninde bu dizilim sayılarının toplamını tutalım.

k değişkenini ise sonuç dizisinin gözlerine ulaşmak için kullanalım, ilk değeri 1 olsun.

1 milyonun içinde bulunan i! sayısı k'ıncı gözde bulunan sayıyı verecektir. Örneğin 1 milyonun içinde 2 tane 9! + 7 tane 8! vardır. Buradan 1. göz 2, 2. göz 7 olarak bulunur. İlk 9 gözü doldurmak için aşağıdaki sözde kod kullanılabilir.

while i > 0 do  
begin  
 kacTane = 0  
 while toplam < 1000000 do  
 begin  
 toplam = toplam + factorial(i)  
 kacTane++  
 end  
 toplam = toplam - factorial(i)   
 kacTane = kacTane -1  
 i = i -1  
 sonuç[k] = sayılar[kacTane]  
 k = k + 1  
 for index = kacTane to 9 do  
 begin  
 sayılar[index] = sayılar[index + 1]  
 end  
end

Yukarıdaki while döngüsünde 1 milyon geçildiği için geçilen miktar geri çıkarılıyor.

Burada kullanılan sayı bir daha kullanılmaması için dizinin elemanları o sayının üzerine doğru 1 geriye kaydırılıyor.

0'dan 9'a kadar olan sayıların toplamı 45'tir. Kullanılmayan son sayı 10. göze atanacaktır. Bunun için ilk 9 gözdeki sayılar toplanır ve 45'ten çıkarılır.

sonuç dizisinin elemanları yazdırılır.

**Problem 26**

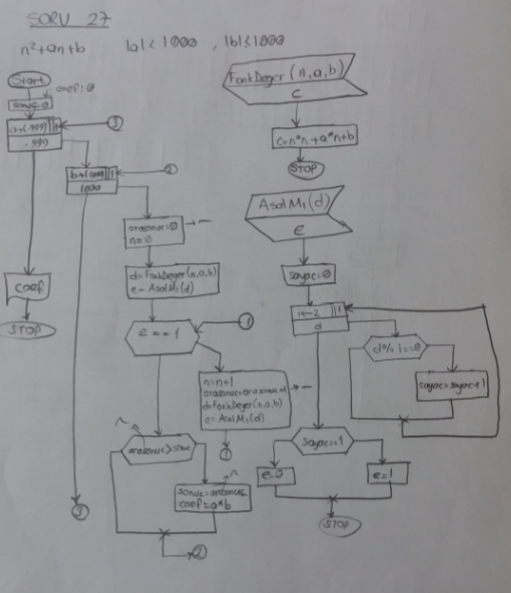
Problemde ilk 1 / i, 1 < i < 1000 sayı içerisinde virgülden sonra devir sayısı en çok olan i değeri isteniyor. Bunun için gerçek hayatta elle çözdüğümüz bölme işlemlerinde uyguladığımız işlemleri yapacağız. İçine gönderilen sayının virgülden sonraki devreden basamak sayısını geri döndüren recurring\_cycle fonksiyonunu oluşturalım.

Fonksiyon içerisinde control değişkeni true olduğu sürece çalışan bir do-while döngüsü kullanılsın. control değişkeninin ilk değeri true olsun. Bölme işleminin virgülden sonraki kısmını result isminde string bir değişkende tutalım. remainder her adımda bulunan kalanı, remainders bağlı listesinde ise bulunan kalanları tutalım. remainders bağlı listesinin ilk elemanını 1 yapalım, çünkü 1 mod i 1'e eşittir. Bölünecek sayı, yani 1'i divided değişkeninde depolayalım.

Koşulu control olan do-while döngüsüne girildiğinde divided 10 ile çarpılsın, çünkü bölme işlemi yaparken kalan sayı 0 değilse 10 ile çarpılır ve tekrar bölen'e bölünür. İçeride bir while döngüsünde divided i’den küçük olduğu sürece divided değişkenini 10 ile çarpalım ve result string değişkeninin sonuna 0 ekleyelim. Döngüden çıktıktan sonra remainder değişkenine divided mod i değerini atayalım. result değişkeninin sonuna divided / i tam sayı bölmesinin sonucunu ekleyelim ve divided değişkenine remainder değerini atayalım. Eğer remainders bağlı listesi remainder değerini içeriyorsa control ifadesini false yapalım, eğer içermiyorsa remainders bağlı listesine remainder değerini ekleyelim ve remainder 0 ise control değişkenini false yapalım. Bu döngü kalan 0 ya da daha önce karşılaşılan bir kalan bulunana kadar dönecektir.

Döngüden çıkıldığında fonksiyon eğer remainder 0 ise 0, değilse result string ifadesinin karakter sayısı ile remainders bağlı listesinde remainder değerinin indisi farkını döndürür. Örneğin 1/6, 1.666... devreden değerine sahiptir, döngüden çıkıldığında result ifadesinin uzunluğu 2 olacaktır, remainder ise 4'tür. remainder ifadesinin bağlı listedeki yeri, yani ilk geçtiği yerin indisi ise 1'dir. (İndislerin 0'dan başladığı kabul edilmiştir.) Burada string uzunluğu 2, indis 1 olduğuna göre 2-1 = 1 devir sayısına sahiptir.

**Problem 27**

****

**Problem 28**

Ortadaki değer 1 olduğu için sonucu tutacak sum değişkeni 1 ile ilklendirilir. İncelendiğinde değerler 4 köşeyi dolaşıp sum’a ekleniyor. Örneğin 3, 5, 7, 9, 13, 17, 21, 25... gibi. Her dörtte bir merkezden 1 uzaklaşıldığı için çap 2 artıyor, ve artış miktarı da 2 artıyor, başlangıçta 3, 5 gibi artış miktarı 2 iken, bir dış spiralde 17, 21 gibi artış miktarı 2 artarak 4 oluyor. Her spiral sağ üst köşede o zamana kadarki kenar uzunluklarının çarpımı değerinde bitiyor, örnekte verilen spiral 5x5 uzunlukta olduğundan 25'te bittiği gibi istenen uzunlukta da 1001x1001 sayısında spiral bitmelidir. Buna göre i değişkeni 1, ve artış miktarını belirleyen increment değişkeni 0'a ilklendirilir.

While döngüsü açılır. İçeride de 1'den 4'e for döngüsü açılır ve sum değeri for döngüsünde i kadar artırılır ve i değeri her iterasyonda increment kadar artırılır. For döngüsünden çıkıldığında increment değişkeni 2 artırılır ve while döngüsü yeni bir iterasyon yapar.

Eklenen i sayısı 1001'in karesi olduğunda döngü durdurulur, sum değeri ekrana bastırılır.

**Problem 29**

distinctPower isimli bir bağlı liste oluşturulur. a indisli 2'den 100'e kadar bir for döngüsü açılır, içine b indisli 2'den 100'e kadar bir for döngüsü açılır. Eğer distinctPower bağlı listesinde ab yoksa listeye eklenir. Döngülerden çıkıldığında distinctPower bağlı listesinin eleman sayısı istenilen sayıdır.

**Problem 30**

Problemde bir sayının basamaklarının 5. kuvvetlerinin toplamı o sayıya eşit olan sayıların toplamı isteniyor.

Bu sayı en fazla 6 basamaklı olabilir, çünkü bütün basamaklar 9 olsa bile 95\*6 354294 ediyor, sonuç 6 basamaklı sayıları geçemediğinden 6 basamaklı sayılara kadar kaba kuvvet yöntemi ile denenerek istenen sayılar elde edilebilir.

Bunun için 10'dan 354294'e kadar for döngüsü açılır, sayı 10'a bölüne bölüne 0 oluncaya kadar bir while döngüsünde kalanlarının 5. kuvveti alınarak ve bulunan değer her basamakta basamakToplamı değişkeninde toplanır, bu değer sayıya eşitse toplam isimli değişkene sayı eklenir.

Bulunan değer ekrana yazdırılır.

**Problem 32**

Öncelikle sum\_of\_pandigital\_products değişkeni 0'a ilklendirilir, pandigitalProducts integer sayılar depolayacak bir bağlı liste oluşturulur. Çarpan, çarpılan ve sonuç toplam 9 basamaktan oluşacağı için çarpan ve çarpılan en fazla 4 basamaklı olabilir. 1'den 9999'a kadar indisi i olan bir for döngüsü açılır. Döngünün içerisinde i+1'den 9999'a kadar indisi j olan bir for döngüsü daha açılır. product değişkeni i\*j'ye eşitlenir. Sayılar basamaklarına ayrılarak karakter dizilerine dönüştürülür ve i, j, product karakter dizileri birbirine eklenir, eklenme sonucu oluşan dizi result olsun. result dizisinin karakter sayısı 9 ise result dizisi isPandigitalWithLength9 fonksiyonuna gönderilir. Burada sıralanır, ve 123456789 dizisiyle karakterleri aynı sırada olup olmadığı kontrol edilir. Eğer aynı sırada ise result bir pandigital sayıdır ve fonksiyon true döndürür, aynı sırada değilse fonksiyon false döndürür. Eğer fonksiyon true döndürürse product, pandigitalProducts bağlı listesine eklenir.

Döngülerden çıktıktan sonra pandigitalProductsOnlyOne isimli bir başka bağlı liste oluşturulur, çünkü pandigitalProducts bağlı listesinde birden fazla aynı product bulunabilir, for döngüsü ile pandigitalProductsOnlyOne listesine eğer bu listede yoksa pandigitalProducts listesinden sırayla elemanlar eklenir, sum\_of\_pandigital\_products eklenen elemanlar ile toplanır. Döngüden çıkıldığında sum\_of\_pandigital\_products değişkeninde istenen değer depolanmış olacaktır. Bu değer ekrana yazdırılır.

**Problem 33**

int main () {  
 int i , j , k , num\_pro , denum\_pro,sonuc;  
 num\_pro = 1;  
 denum\_pro = 1;  
 for (i=1 ; i<10 ; i++) {  
 for (j=1 ; j<10 ; j++) {  
 for (k = 1 ; k<10 ; k++) {   
 if (((10\*i+j)\*k) == ((10\*j+k)\*i)) {  
 num\_pro \*=i;  
 denum\_pro \*=k;  
 }  
 }  
 }  
 }  
 sonuc = LowTermDen(num\_pro,denum\_pro);  
 printf("Sonuc : %d",sonuc);  
 return 0;  
}  
int LowTermDen(int a , int b) {  
 int l , c , d;  
 l = 3;  
 c = a;  
 d = b;  
 while ((c % 2 == 0) && (d % 2 == 0)) {  
 c = c/2;  
 d = d/2;  
 }  
 while ((l<=c) && (l<=d)) {  
 while ((c % l == 0) && (d % l == 0)) {  
 c = c / l;  
 d = d / l;  
 }   
 l = l + 2;   
 }  
 return d;  
}

**Problem 34**

Problemin çözümünü hesaplamadan önce bu sayıların en büyüğünün en çok kaç olabileceği bulunmalıdır.

Bütün basamakları 9 olduğu durumda sayının basamaklarını faktöriyelleri toplamı 9!\*n olacaktır. Sayının değeri ise 9 \* (1 + 10 + ... + 10n-1) olacaktır.

9\*(1+10+...+10n-1) = 9 \* (10n-1+1-1) / (10-1) = 10n-1

Sayının değeri bu şekilde bulunduktan sonra yapılması gereken n'in en büyük değerini bulmaktır, yani 9! \* n < 10 ^ n - 1 eşitsizliğini çözmektir. Burada n'in alabileceği en büyük değer yaklaşık 6.5 olarak bulunur. (WolframAlpha ile hesaplandı.) Yani istenen sayılar en fazla 6 basamaklı olabilir.

sum isimli bir değişken oluşturup 0'a ilklendirilir. 10'dan 1 milyona kadar bir for döngüsü çalıştırılır. Döngünün indisi i olsun. Döngü içersinde sum\_of\_digits\_factorial değişkene 0 ve tmp\_num isimli değişkene i atanır. tmp\_num 0'dan farklı olduğu sürece dönen bir while döngüsünde tmp\_num mod 10 ile son basamak alınır, bu basamağın faktöriyeli hesaplanıp sum\_of\_digits\_factorial’e eklenir. tmp\_num 10'a bölünür. Bu şekilde while döngüsünde sayının bütün basamaklarının faktöriyelleri toplamı sum\_of\_digits\_factorial değişkeninde depolanmış olur. Bundan sonra ise i, sum\_of\_digits\_factorial’e eşitse sum değişkenine i eklenir. For döngüsünden çıkıldığında elde edilen sum istenilen değerdir.

**Problem 35**

Burada öncelikle 1 milyon elemanlı bir boolean dizi açılıp her elemanına true atanır. 1'den 1 milyona kadar while döngüsü ile bakılır, ve bakılan eleman true ise primes bağlı listesine eklenir, o elemanın dizideki tam katları false yapılır. Sonra primes bağlı listesi üzerinde while döngüsü ile gezilir, elemanlar basamaklarına 10'a bölünerek ayrılır ve basamaklarında çift sayı veya 5 bulunan elemanlar listeden silinir, 2 ve 5 hariç. Bu silme işlemi basamaklar kaydırıldığında, çift sayı veya 5 olan basamaklar birler basamağına geldiğinde sayının circularPrime olmasını engellememesi için yapılır. Sonrasında bağlı listedeki elemanlara daha hızlı ulaşmak için bunlar primesElected dizisine atılır ve circularPrimes isimli bir bağlı liste oluşturulur.

For döngüsü ile primesElected dizisinde baştan sona gidilir. Döngünün indisi i olmak üzere, her iterasyonda circularPrimes bağlı listesinde i'inci indise ait elemanın olup olmadığına bakılır, varsa bir sonraki indise geçilir. Diğer durumda numOfDigits isimli değişkene i'inci elemanın basamak sayısı depolanır. tmp\_circulars isimli boyutu numOfDigits olan bir dizi oluşturulur ve ilk değeri i'inci indise ait elemandır. Sonra i'inci indise ait elemanın basamakları kaydırılmış hallerinin de asal sayı olup olmadığı binary search ile primesElected dizisinde aratılır. Burada basamakları kaydırmak için nextCircle isimli fonksiyon kullanılır. Eğer kaydırılmış basamak değerli sayı asal ise tmp\_circulars dizisine eklenir. tmp\_circulars dizisine eğer basamak sayısı kadar eleman eklendiyse bu i'inci indise sahip asal sayının bir circular prime olduğu anlamına gelir ve tmp\_circulars dizisinde halkadaki diğer asal sayılar da vardır. tmp\_circulars dizisindeki elemanlar circularPrimes bağlı listesine kopyalanırlar.

circularPrimes bağlı listesinin elemanları içerisinde birbirinin aynı iki eleman olabilir, örneğin 11 sayısı basamak kaydırıldığında da kendisine eşittir, bu listedeki elemanlar da başka bir listeye her birinden bir tane olacak şekilde kopyalanır. Kopyalanan listenin eleman sayısı istenen değerdir.

nextCircle fonksiyonu içine aldığı sayının 10'a bölümünden kalanını tutar. Sayıyı 10'a tam sayı bölmesi yapar. Sonra da kalanı "10 üzeri basamak sayısı -1" ile çarparak sayıya ekler. Bu sayede basamaklar bir sağa kaymış olur.

**Problem 36**

Problemde hem on, hem iki tabanında palindrom sayı olması istendiği için sayılar tek olmalıdır, çünkü sadece sonu tek olan sayıların 2 tabanında eşitleri 1 ile biter, bir sayı 0 ile başlayamayacağı için sonu 0 olan sayılar palindrom olamaz.

1'den 1000000'a kadar ikişer ikişer artan bir for döngüsü çalıştılır. Her dönüşte sayının 2 tabanında ve 10 tabanında palindrom olması halinde sum değişkenine sayı eklenir. Döngü bittiğinde sum değişkeninin tuttuğu değer problemde istenen sayıdır.

Döngünün içinde 2 ve 10 tabanında sayılarının palindrom olup olmadığının kontrolü ise bir fonksiyon ile yapılır. Fonksiyon parametre olarak string bir ifade içinde sayıyı alır ve yarısına kadar çalıştırılan bir döngüde i. ve n-i+1. elemanların birbirine eşit olup olmadığı kontrol edilir, eşit olmayan eleman çıkarsa false çıkmazsa true döndürülür. Burada n string'in uzunluğu yani 2 ya da 10 tabanındaki sayının basamak sayısıdır. Fonksiyonu kullanabilmek için ise sayıyı 0 olana kadar 10'a ve 2'ye bölerek elde edilen kalanlardan string ifadeler elde edilir.

**Problem 37**

generatePrimes isimli bir fonksiyon oluşturulup 1 milyona kadar olan asal sayılar bir diziye atılır. sum değişkeni 0 olarak ilklendirilir. Asal sayılar dizisinin 7'den sonraki asal sayıları üzerinde for döngüsü ile gezilir, isTruncatable fonksiyonuna her iterasyonda bir asal sayı gönderilir ve fonksiyon true döndürürse bu sayı sum değişkenine eklenir. Döngüden çıkıldığında elde edilen toplam yazdırılır.

isTruncatable fonksiyonu asal sayı ve asal sayıların bulunduğu diziyi parametre olarak alır. İçeride while döngüsünde asal sayı 0 olana kadar 10'a tam sayı bölmesi yapılır ve her sonuç asal sayılar dizisinde aranır, bulunamazsa fonksiyon false döndürür, bulunursa döngü devam eder. Bu şekilde sayıya sağdan sola yaklaşılmış olur, eğer her sayı dizide bulunursa döngüden çıkılır, fonksiyon devam eder. Bir for döngüsü açılır, 1'den asal sayının basamak sayısına kadar döner, döngünün indisi i olsun. Her iterasyonda sayı 10 üzeri i'ye bölünür ve sonuç asal sayılar dizisinde var mı kontrol edilir, yoksa fonksiyon false değerini döndürür. Bu şekilde sayıya soldan sağa yaklaşılmış olur. For döngüsü boyunca bütün sayılar dizide bulunursa döngü true döndürür.

**Problem 38**

Soruda istenen sayı 5000 ile 9999 arasında bir sayı olmalıdır, çünkü bu sayıların bir katları ile iki katları birbirine bağlandığında 9 basamaklı bir sayı oluşur.

9999'dan 5000'e azalan bir for döngüsü açılır. Döngünün içerisinde pandigitalMultiples isimli karakter tipinde değerler tutacak bir bağlı liste oluşturulur. İki farklı karakter dizisine, sayı ve sayının iki katı basamaklarına ayrıştırılarak atılır. Bu diziler bağlı listeye eklenir. Bir sonuc karakter dizisi oluşturulur ve bağlı listedeki elemanlar bu diziye kopyalanır. isPandigital fonksiyonuna bu dizinin bir kopyası gönderilir. Burada dizi sıralanır ve sıralı dizideki elemanlar 1, 2,..., 9 sırasına sahip olup olmadıkları kontrol edilir, eğer sahipse true, değilse false döndürülür. Fonksiyondan dönen değer true ise sonuc dizisi ekrana yazdırılır ve program durdurulur.

**Problem 39**

int p , a, b, maxsol , sol , maxp,c;  
maxsol = 0;  
for (p=1 ; p<=1000 ; p++) {  
 sol = 0;  
 for (a=1 ; a<=(int)((float)p/sqrt(2)) ; a++ ) {  
 for (b=1 ; b<=(int)((float)p/sqrt(2)) ; b++) {   
 c = p - a - b;  
 if (c\*c == a\*a + b\*b) {  
 sol++;  
 }  
 }  
 }  
 if (sol > maxsol) {  
 maxp = p;  
 maxsol = sol;  
 }  
}  
printf("1000'den kucuk maksimum cozumum oldugu p sayisi ve cozum sayisi : %d / %d",maxp,maxsol);

**Problem 40**

İstenen basamaklar arasında 1 milyonuncu basamak da olduğu için öncelikle kaçıncı sayıya gelindiğinde 1 milyon basamaktan büyük eşit sayıda basamak oluştuğu bulunmalıdır, bunun için i 1'e, number 0'a ilklendirilirse,

i basamak sayısı olmak üzere, i basamaklı 9 \* 10i-1 sayı vardır, bu değer i ile çarpılarak ve number’a eklenerek i'inci basamağa kadar oluşan toplam basamak sayısı bulunabilir.

while (number <= 1000000) {  
 number += i \* 9 \* Math.pow(10, i - 1);   
 i++;  
}

while döngüsü sonucunda elde edilen i-1 değeri istenen basamakların ilk 10i-1 sayı içinde olduğunu gösterir.

fraction isimli bir karakter tipinde elemanlar depolayacak bir bağlı liste oluşturulur. 1'den 10i-1'e kadar for döngüsü çalıştırılır ve döngünün içinde şu işlemler yapılır:

Döngünün indisi j olsun.

j'inci sayı basamaklarına ayrılır ve basamaklar bir karakter dizisine atılır, bu dizideki karakterler fraction bağlı listesine eklenir.

Son olarak istenen basamaklara bağlı listeden erişilir ve istenen çarpım yapılarak sonuca ulaşılır.

**Problem 41**

number değişkeni 999999999'a ilklendirilir, while döngüsü içerisinde ikişer azalarak iterasyon yapar. Döngü içerisinde iç içe iki koşul vardır, birinci koşulda sayı isPandigital fonksiyonu ile pandigital olup olmadığı kontrol edilir, eğer öyleyse ikinci koşulda isPrime fonksiyonu ile asal olup olmadığı kontrol edilir, asalsa döngü durur. değilse iki azalarak devam eder.

isPandigital fonksiyonunda numerals isimli string tipinde bir değişken "123456789"a ilklendirilir. Fonksiyona parametre olarak gelen number basamaklarına ayrılarak bir karakter dizisine dönüştürülür, bu dizinin ismi num olsun. Dizi sıralanır ve dizinin uzunluğuna kadar numerals ve num değişkenlerinin elemanları aynı mı diye bakılır, aynıysa fonksiyon true eğer farklı karakter varsa fonksiyon false değerini döndürür.

isPrime fonksiyonunda number parametre olarak alınır. 3'ten i^2 küçük eşit number olana dek bir while döngüsü çalıştırılır ve döngüde number mod i 0'a eşitse fonksiyon false, eğer döngüde hiçbir number mod i değeri 0'a eşit olmadan çıkılırsa fonksiyon true değerini döndürür.

**Problem 42**

Öncelikle ilk 35 tane üçgensel sayıyı içeren üçgenselSayılar isimli bir dizi oluşturuldu. adet isimli istenen kelimelerinin sayısının depolanacağı değişken de 0'a ilklendirildi.

Kelimeler bir diziye alındı. For döngüsü ile kelimelerin bulunduğu dizide her kelimeye ulaşıldı. Burada toplam isimli kelimelerin toplam karakter değerlerini taşıyacak bir değişken 0'a ilklendirildi. İçeride bir for döngüsü daha çalıştırılarak kelimelerin bütün karakterlerine ulaşıldı. Kelimeler karakterlerden oluştuğu için ve karakterlerin sayı karşılığı da ASCII sayıları olduğu için her harfte toplam değişkenine aşağıdaki değer eklendi.

toplam = toplam + kelimeler[i][j] - 'A' + 1

Burada yapılan işlem ilk harf A olduğu için ASCII değerlerinden A çıkarıp 1 eklemek. Örneğin B ikinci harf ve ASCII karşılığı 66, A'nın ASCII karşılığı ise 65, yapılan işlemde 66 - 65 + 1 yapılarak 2 sonucu elde ediliyor. Bu şekilde bütün kelimelerin harf değerleri hesaplanıyor.

Harf değeri hesaplanan kelimelenin harf değerinin üçgensel sayı olup olmadığı ise üçgenselSayılar dizisinde binarySearch ile aratıldı, eğer sonuç bulunduysa adet değişkeni 1 artırıldı.

adet değişkeni ekrana yazdırıldı.

**Problem 43**

Bu soruyu çözmek için öncelikle 10 basamaklı bütün pandigital sayılar üretilmelidir, bunu yapmak için permütasyon kullanılmalıdır, bunun için aşağıdaki hazır algoritma kullanılmıştır.

1-Find the largest index k such that a[k] < a[k + 1]. If no such index exists, the permutation is the last permutation.  
2-Find the largest index l greater than k such that a[k] < a[l].  
3-Swap the value of a[k] with that of a[l].  
4-Reverse the sequence from a[k + 1] up to and including the final element a[n].  
  
Kaynak: https://en.wikipedia.org/wiki/Permutation#Algorithms\_to\_generate\_permutations

Her permütasyon ile yeni bir sayı üretildikten sonra substring\_divisibility fonksiyonuna gider ve sayının problemde verilen kurala uygun olup olmadığı kontrol edilir, uygun olan sayılar toplanır ve ekrana yazdırılır.

substring\_divisibility fonksiyonunda nums isimli bir dizi oluşturulup bu dizide sırasıyla soruda verilen, tam bölünüp bölünmediği test edilecek sayılar yerleştirilmiştir. i bu dizinin indisidir. j 6 ile ilklendirilmiştir, çünkü ilk kontrol edilecek basamaklar 2,3,4. basamaklardır, ve sayı 10 basamaklı olduğundan, 10 üzeri 6 ile bölünüp, 10 üzeri 3 ile kalanı alınırsa istenilen basamaklar elde edilecektir. Bir while döngüsü açılır ve i 7'den küçük ve sayılar tam bölündüğü sürece döner. Döngünün içinde sayı 10 üzeri j'ye bölünüp, 10 üzeri 3 ile kalanı alınarak nums[i]'ye bölümünden kalana bakılır, 0 değilse döngü durur ve fonksiyon false döndürür. 0'sa i 1 artar, j bir azalır ve kontroller devam eder.

**Problem 44**

İlk 10000 pentagonal sayı for döngüsü ile bir diziye atıldı. i ve j, sırasıyla dışarıdaki ve içerdeki for döngüsü olmak üzere iç içe iki tane for döngüsü açıldı. Dış döngü 0'dan 9999'a, iç döngü i+1'den 9999'a gönderildi, iç döngüde dizinin i ve j'inci elemanları toplandı. Bu toplam pentagonal bir sayı mı diye binarySearch ile dizide araştırıldı, aynı şekilde çıkarılarak binarySearch ile dizide araştırıldı, hem toplam hem de fark pentagonal sayılar dizisinde bulunduğunda istenilen fark değeri ekrana yazdırıldı.

**Problem 45**

int main () {  
 long int i , result;  
 i = 144;  
 result = Hexa(i);  
 while (isPenta(result,i) != 0) {  
 i++;  
 result = Hexa(i);  
 }  
 i = 2\*i - 1;  
 printf("Bulunan triangle, hexagonal ve pentagonal sayi : %ld\n", result); return 0;  
}  
long int Hexa(long int a) {  
 long int b;  
 b = a \* (2\*a-1);  
 return b;  
}  
long int Penta (long int c) {  
 long int d;  
 d = c \* (3\*c - 1) / 2;  
 return d;  
}  
long int isPenta(long int x , long int f) {  
 long int r;  
 r = x - Penta(f);  
 while (r > 0) {  
 f++;  
 r = x - Penta(f);  
 }  
 return r;  
}

**Problem 46**

Problemde "Goldbach's other conjecture" olarak nitelendirilen kural bütün tek kompozit sayıların, bir asal ve bir sayının karesinin iki katına eşit olduğu ama bu kuralın yanlış olduğu söyleniyor, bu kuralı bozan ilk kompozit sayı isteniyor.

Bunun için 12. sorudaki generatePrimes fonksiyonu kullanılmıştır, primes dizisine ilk 10000 asal sayı üretilmiştir, bu üretim yapılırken kullanılan numbers dizisinde ise sadece kompozit sayıların indisi 0 olarak kaldığından composites isimli tek kompozit sayıları barındıran dizi numbers dizisi kullanılarak oluşturulabilir. Bunun için numbers dizisi üzerinde bir for döngüsü çalıştırılır ve numbers dizisinin her iterasyonunda 0'a eşit olan ve tek olan sayıları composites dizisine eklenir.

Bir while döngüsü çalıştırılır ve her iterasyonda isGoldbachComposite fonksiyonu çalıştırılır, fonksiyona composites dizisindeki sayılar sırayla yollanır, eğer fonksiyon false değeri döndürürse döngü durur. False değeri dönmesine sebep olan sayı istenen sayıdır.

isGoldbachComposite fonksiyonu ise şu şekilde çalışır: Bakılacak kompozit sayı ve asal sayıların bulunduğu diziyi parametre olarak alır. Asal sayılar dizisindeki sayı kompozit sayıdan küçük olduğu sürece bir while döngüsü çalıştırılır, döngüde ise

karekok((kompozit sayı - asal sayı) / 2)

ifadesinin tam sayı olup olmadığı kontrol edilir. Bunun için ifade en yakın tam sayıya yuvarlanır ve yuvarlanmış ifade, ifadeye eşit mi bakılır. Eğer tam sayı ise döngü durur, fonksiyon true döndürür, eğer döngüden çıkılana kadar böyle bir durumla karşılaşılmazsa fonksiyon false döndürür.

**Problem 47**

Problemde art arda gelen 4 tane 4 asal çarpana sahip ilk sayı isteniyor.

Bunun için problem 12'deki generatePrimes fonksiyonu ile ilk 10000 asal sayıyı içeren primes dizisi üretilmiştir. count değişkeni 0, number değişkeni 2 olarak ilklendirilmiştir. Sonra da count 4'ten küçük olduğu sürece çalışan bir while döngüsü açılmıştır. While döngüsü içinde eğer number’ın asal çarpan sayısı 4 değilse count 0'a eşitlenir, değilse count 1 artırılır. number her iterasyonda bir artırılır. count 4 olup döngüden çıkıldığında ise elde edilen number’ın 4 eksiği istenen sayıdır.

Asal çarpan sayısını kontrol ederken number\_of\_prime\_factors isimli fonksiyona o iterasyondaki number gönderilir, burada da number’ın yarısına gelene kadar döngü içerisinde number’ın primes dizisindeki asal sayı ile tam bölündüğü durumda primeFactors bir artırılır ve fonksiyondan geriye primeFactors döndürülür.

**Problem 48**

* toplam isimli bir değişkeni 0'a ilklendirilir.
* 1'den 1000'e kadar for döngüsü çalıştırılır. Döngünün indisi i olsun.
* Her dönüşte toplam'ın değer ii kadar artırılır.
* Ekrana toplam’ın 1010'a bölümünden kalan yazdırılır.