42 C05 Çözümleri

ex00:

ft_iterative_factorial Fonksiyonu:

Bu fonksiyon, verilen pozitif bir tam sayının faktöriyelini hesaplar. Eğer verilen sayı negatifse, 0 döndürülecektir.

İşleyişi:

- Negatif Sayılar: Eğer nb negatifse, fonksiyon 0 döndürür.
- Pozitif Sayılar: Eğer nb sıfır veya pozitifse, 1'den nb'ye kadar olan tüm sayıları çarparak faktöriyel hesaplanır.

Kullanım:

- Bu fonksiyon, döngüsel bir yöntemle faktöriyel hesaplamaktadır. Faktöriyel, sayıdan birer birer geriye giderek çarpılmak suretiyle hesaplanır.

Faktöriyel hesaplamak için iterative (döngüsel) fonksiyon :

```
#include <stdio.h>

int ft_iterative_factorial(int nb)
{
    int result = 1;
    if(nb < 0)
        return (0);

    while(nb > 0)
    {
        result *= nb;
        nb--;
    }
    return result;
}

int main()
{
    printf("Sonuç : %d \n",ft_iterative_factorial(5));
    return (0);
}
```

ex01
Kısaca, yine faktöriyel hesapla ama rekürsif olsun diyor.

Rekürsif: Fonksiyon kendi kendini çağırır ve problemi küçük parçalara böler. Sonunda bir temel duruma ulaşır ve çözüm bulunur.

ft_recursive_factorial Fonksiyonu:

- Eğer nb negatifse, 0 döndürülür.
- Eğer nb 0 veya 1 ise, fonksiyon 1 döndürür (bu, faktöriyel hesaplamada temel durumdur).
- Diğer durumlarda, faktöriyel nb * ft_recursive_factorial(nb - 1) şeklinde hesaplanır. Bu, rekürsif bir çağrıdır.

```
#include <stdio.h>
int ft_recursive_factorial(int nb)
{
    if(nb < 0)
        return (0);

    if(nb == 0 || nb == 1)
        return (1);

    return nb* ft_recursive_factorial(nb -1);
}

int main()
{
    printf("Sonuç : %d \n",ft_recursive_factorial(5));
    return (0);
}
```

ex02
Bu örnek’te ise, bir sayının küvvetini (üst’lü sayı hesaplama)

```
#include <stdio.h>
int ft_iterative_power(int nb, int power)
{
    int i = 0;
    int result = 1;

    if(power < 0)
        return (0);

    if(power == 0)
        return (1);

    while(i < power)
        {
            result *= nb;
            i++;
        }
    return (result);
}
```

```
int main ()
{
    printf("Sonuç : %d\n",ft_iterative_power(10,3));
    return (0);
}
```

Bu fonksiyon, bir sayının (nb) üssünü (power) iteratif olarak hesaplar

- Eğer power < 0 ise **0 döndürür** (negatif üs desteklenmiyor).
- Eğer power == 0 ise **1 döndürür** (her sayının 0. kuvveti 1'dir).
- while döngüsüyle nb, power kadar kendisiyle çarpılarak result değişkeninde saklanır.
- Son olarak hesaplanan değer döndürülür.

ex03

Mantık Aynı Yine bir sayının üstünü alıcağız fakat rekürsif fonksiyon kullanarak.

```
#include <stdio.h>
int ft_recursive_power(int nb, int power)
{
    if(power == 0)
        return (1);

    if(power < 0)
        return (0);

    return (nb * ft_recursive_power(nb, power - 1));
}
int main()
{
    printf("Sonuç : %d\n",ft_recursive_power(2,3));
    return(0);
}
```

- Eğer power == 0 ise herhangi bir sayının 0. kuvveti 1 olduğu için 1 döndürülür.
- Eğer power < 0 ise negatif kuvvetleri işleyemediğimiz için 0 döndürülür.

Recursive Adım

- nb * ft_recursive_power(nb, power - 1) ifadesiyle fonksiyon kendisini çağırarak kuvvet işlemi yapılır.
- Kuvvet her seferinde power - 1 olacak şekilde azalır ve sonunda power == 0 olduğunda recursion durur.

ex04

Fibonacci’nin ne olduğunu öğrenelim fibonacci sayı dizisinde bir öncekinin toplamıdır bir sonraki adım,

0, 1, 1, 2, 3, 5, 8

0 + 1 = 1

1 + 1 = 2

1 + 2 = 3

2 + 3 = 5

3 + 5 = 8

```
#include <stdio.h>
int ft_fibonacci(int index)
{
    if(index < 0)
        return(-1);

    if(index == 0)
        return (0);

    if(index == 1)
        return (1);

    return (ft_fibonacci(index -1) + ft_fibonacci(index -2));
}

int main ()
{
    printf("Sonuç : %d\n",ft_fibonacci(10));
    return (0);
}
```

Fonksiyon Başlangıcı: ft_fibonacci fonksiyonu, bir index alır ve bu index'in Fibonacci dizisindeki karşılık gelen sayısını döndürür. Bu fonksiyon, Fibonacci dizisini hesaplamak için üç temel adımı izler.

- Hatalı Giriş Kontrolü: Eğer index negatif bir değer alırsa, program hatalı bir durum olduğunu belirterek -1 döndürür. Fibonacci dizisi yalnızca pozitif indeksler için geçerlidir.
- Temel Durumlar: Fibonacci dizisinin iki temel durumu vardır

  Fibonacci(0) = 0
  Fibonacci(1) = 1 Eğer index 0 veya 1 ise, doğrudan bu değerler döndürülür.
- Rekürsif Hesaplama: Eğer index 0 veya 1 değilse, program Fibonacci sayısını hesaplamak için rekürsif (kendini tekrar eden) bir yaklaşım kullanır. Bu, index - 1 ve index - 2'nin Fibonacci sayılarının toplamını döndürerek yapılır. Bu sayede, daha küçük indeksler için Fibonacci sayıları hesaplanır ve geri doğru sonuçlar elde edilir.

ex05

Bir sayının karekökünü çıktı olarak veren bir fonksiyon yazmamız isteniyor

```
#include <stdio.h>
int ft_sqrt(int nb)
{
    int i = 0;

    if (nb < 0)
        return (0);

    while (i * i <= nb)
    {
        if (i * i == nb)
        {
            return (i);
        }
       i++;
    }
    return (0);
}

int main()
{
    printf("Sonuç : %d \n",ft_sqrt(144));
    return (0);
}
```

ft_sqrt fonksiyonu, bir tam sayı alır ve eğer sayı negatifse 0 döndürür. Ardından, 0'dan başlayarak karekökünü bulmaya çalışır. Her seferinde i * i ifadesiyle karesini kontrol eder. Eğer i * i verilen sayıya eşitse, i değeri döndürülür. Döngü sona erdiğinde, sayı bir kare sayı değilse 0 döndürülür

ex06

Asal sayı, yalnızca 1 ve kendisiyle tam olarak bölünebilen, yani tam bölünebilen sadece iki sayı olan pozitif tam sayıdır. Yani, asal sayının sadece iki pozitif böleni vardır: 1 ve kendisi. Bu özellik, asal sayıları diğer sayılardan ayıran temel farktır.

2 sayısını özel olarak kontrol ettik (çünkü 2 asal bir sayıdır ve tek çift asal sayıdır)

- 2'den büyük çift sayılar için hemen 0 döndürdük (çift asal sayılar yoktur, sadece 2 asal sayıdır)
- Döngüde, sadece tek sayılar üzerinde işlem yapıyoruz (i += 2 ile).

Bizden istenen ise, asal sayılar 1 asal sayı değilse 0 olarak girmemiz isteniyor.

```
#include <stdio.h>
int ft_is_prime(int nb)
{
    if(nb < 1)
        return (0);

    if(nb == 2)
        return (1);

    if(nb % 2 == 0)
        return (0);

    int i = 3;
    while (i * i <= nb)
    {
        if (nb %i == 0)
            return (0);
        i += 2;
    }
    return (1);
}

int main()
{
    printf("Sonuç : %d \n",ft_is_prime(5));
    return (0);
}
```