```python
def f_6(n: int):
    for i in range(n):
        t=1
        while t<n:
            print("Hello world")
            t*=2
```

$$T(n)=\sum_{i=0}^{n-1}\log_2 n = \log_2 n \sum_{i=0}^{n-1} 1$$

$$= \log_2 n \underbrace{(1+1+\dots+1)}_{n}$$

$$= n \cdot \log_2 n$$

```python
def f_11(n: int):
    s=0
    for i in range(1, n**2 +1):
        j=i
        while j!=0:
            s = s+j-10*j//10
            j//=10
    return s
```

$$T(n)=\sum_{i=1}^{n^2}\log_{10} i$$

$$= \log_{10} 1+\log_{10} 2+\dots+\log_{10} n^2$$

$$= \log_{10}(1\cdot 2\cdot 3\cdot\dots\cdot n^2)$$

$$= \log_{10} n^2!$$

STIRLING APPROXIMATION

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

$$\log_{10} n! \sim \log_{10}\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

1

$\hookrightarrow \log_{10} n! \sim \log_{10} \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$

$\sim \log_{10}\sqrt{2\pi n} + \log_{10}\left(\frac{n}{e}\right)^n$

$\sim \frac{1}{2}\log_{10} 2\pi n + n\log_{10}\frac{n}{e}$

$\sim \frac{1}{2}\log_{10} 2\pi n + n\log_{10} n - n\log_{10} e$

$\log_{10}\sqrt{2\pi n} = \log_{10}(2\pi n)^{\frac{1}{2}}$

$= \frac{1}{2}\log_{10} 2\pi n$

$T(n) = \frac{1}{2}\log_{10} 2\pi n^2 + n^2 \cdot \log_{10} n^2 - n^2 \log_{10} e$

$= \frac{1}{2}\log_{10} 2\pi + \log_{10} n + 2n^2 \cdot \log_{10} n - n^2 \cdot \log_{10} e.$

$= \underline{2n^2 \cdot \log_{10} n} - n^2 \cdot \log_{10} e + \log_{10} n + \frac{1}{2}\log_{10} 2\pi$

$\frac{1}{2}\log_{10} 2\pi n^2 =$

$= \frac{1}{2}(\log_{10} 2\pi + \log_{10} n^2)$

$= \frac{1}{2}\log_{10} 2\pi + \frac{1}{2}\cdot 2 \cdot \log_{10} n$

$\Downarrow$

$T(n) \in \Theta(n^2 \log_{10} n)$

```
def recursive_f3(n: int):
    if n <= 1:
        return 1
    else:
        return 1 + recursive_f_3(n//2)
```

(EXTENDED SOLUTION)

$T(n) = \begin{cases} 1 & \text{dacă } n \leq 0 \\ T(n/2) + 1 & \text{altfel} \end{cases}$

$T(n) = T(n/2) + 1$

$T(n/2) = T(n/4) + 1 = T(n/2^2) + 1$

$T(n/4) = T(n/8) + 1 = T(n/2^3) + 1$

....

! recursive_f_6.

$T(n) = T(n/2) + 1$

$= [T(n/2^2) + 1] + 1$

$= T(n/2^2) + 2$

$= [T(n/2^3) + 1] + 2$

$= T(n/2^3) + 3$

...

$= T(n/2^k) + k$

$T(n) = T(n/2^k) + k$

$1 \Rightarrow \frac{n}{2^k} = 1 \Rightarrow n = 2^k$

$k = \log_2 n$

$T(n) = T(1) + \log_2 n$

$= 1 + \log_2 n \in \Theta(\log_2 n)$

```
def recursive_f_6(n, i: int):
    if n>1:
        i*=2
        n=n//2
        recursive_f_6(n,i-2)
        recursive_f_6(n,i-1)
        recursive_f_6(n,i+2)
        recursive_f_6(n,i+1)
    else:
        print(i)
```

$T(n) = \begin{cases} 1 \text{ dacă } n \leq 1 \\ 4T(n/2) + 1 \text{ altfel} \end{cases}$

$T(n) = 4T(n/2) + 1$

$T(n/2) = 4T(n/4) + 1 = 4T(n/2^2) + 1$

$T(n/4) = 4T(n/8) + 1 = 4T(n/2^3) + 1$

...

$T(n) = 4T(n/2) + 1$

$= 4[4T(n/2^2) + 1] + 1$

$= 4^2 T(n/2^2) + 4 + 1$

$= 4^2[4T(n/2^3)+1]+4+1$

$= 4^3T(n/2^3)+4^2+4+1$

....

$= 4^k T(n/2^k) + 4^{k-1} + 4^{k-2} + \ldots + 4+1$

$\downarrow 1 \Rightarrow n/2^k = 1$

$n = 2^k$

$T(n) = 4^k \cdot T(1) + 4^{k-1} + 4^{k-2} + \ldots + 4 + 1$

$= 4^k \cdot 1 + 4^{k-1} + \ldots + 4 + 1$

$= 1 + 4 + \ldots + 4^k$

$= \dfrac{4^{k+1}-1}{3} = \dfrac{4n^2-1}{3} \in \Theta(n^2)$

$4^k = (2^2)^k = 2^{2\cdot k} = (2^k)^2 = n^2$