Задача 1: Анализирайте времевата сложност на Alg1

```
ALG_1(A[1...n])
if n < 32
   return 0
if n < 128
   return A[1]
k ← ⌊n/16⌋
p ← k×8
r ← n-k
k ← k+1
x ← A[p]
y ← ALG_1(A[1...k])
z ← ALG_1(A[r...n])
return x+y+z
```

Решение:

Алгоритъмът Alg_1 се извиква рекурсивно два пъти, като всяко от двете извиквания е върху масив с дължина, равна на една шестнадесета от дължината на входния масив.
Останалата работа отнема константно време.

$$\Rightarrow T(n) = 2T\left(\frac{n}{16}\right) + \Theta(1)$$

$$k = \log_{16} 2 = \frac{1}{4}$$

$n^{1/4-\varepsilon} \geq const$ и от първи случай на мастер-теоремата получаваме, че $\boxed{T(n) = \Theta(\sqrt[4]{n})}$.

Задача 2: Анализирайте времевата сложност на Alg_2

```
ALG_2(n)
r ← 1
p ← 4
k ← p
while k ≤ n do
    print k
    k ← 6 × p − 8 × r
    r ← p
    p ← k
```

Решение:

Трасираме алгоритъма Alg_2 и виждаме, че отпечатва следната редица: 4, 16, 64, ...
Тоест алгоритъмът отпечатва числото $k_m = 4^m$ при m-тото изпълнение на тялото на цикъла.

Също така, според псевдокода, в променливата p се пази предишната стойност на k, а в променливата r - по предишната.

Ще докажем, че $k_m = 4^m$ по два начина.

I начин: С математическа индукция:

База: $m=1$: При първото изпълнение на тялото на цикъла отпечатваме $k = 4$.(началната стойност на k), т.е $m_1 = 4^1$
След това k става равно на $6^* 4 - 8^* 1 = 16$ (*)
$m=2$: От (*), при второто изпълнение на тялото отпечатваме $k=16$, т.е. $m_2 = 4^2$.

Инд.предп: Допускаме, че $k_{m-2} = 4^{m-2}$ и $k_{m-1} = 4^{m-1}$ се отпечатват съответно при m-2-рото и m-1-вото изпълнения на тялото на цикъла.

Инд. стъпка: Разглеждаме m-тото изпълнение на тялото.
На това изпълнение отпечатваме стойността на k, която й е била присвоена на m-1-то изпълнение, а именно $k_m = 6^* k_{m-1} - 8^* k_{m-2}$ (p и r бяха предишната и по-предишната ст-ти на k).

От инд. предп. получаваме, че $k_m = 6^* 4^{m-1} - 8^* 4^{m-2} = 4^{m-2}(6.4-8) = 4^{m-2} . 16 = 4^{m-2} . 4^2 = 4^m$, т.е $\boxed{k_m = 4^m}$ ■

II начин: С характеристично уравнение:

Решаваме линейно – рекурентното уравнение $k_m = 6k_{m-1} - 8k_{m-2}$, $m \geq 3$ при начални условия $k_1 = 4$ и $k_2 = 16$.

1) Правим характеристично уравнение:

$$x^m = 6x^{m-1} - 8x^{m-2} \mid : x^{m-2}$$

$$x^2 = 6x - 8$$

$$x^2 - 6x + 8 = 0$$

$$x_{1,2} = 3 \pm \sqrt{9-8} = 3 \pm 1 \quad \begin{matrix} \nearrow x_1 = 4 \\ \searrow x_2 = 2 \end{matrix}$$

2) Мултимн. от корените: $\{2, 4\}_м$.

3) $k_m = A \cdot 2^m + B \cdot 4^m$ ← общо решение

4) Намираме коефициентите A и B:

4.1.) Заместваме в общото решение m с 1 и 2:

$$\left| \begin{aligned} k_1 &= A \cdot 2^1 + B \cdot 4^1 \\ k_2 &= A \cdot 2^2 + B \cdot 4^2 \end{aligned} \right.$$

4.2.) Знаем, че $k_1 = 4$, $k_2 = 16$, така че:

$$\left| \begin{aligned} 4 &= 2A + 4B \mid : 2 \\ 16 &= 4A + 16B \mid : 4 \end{aligned} \right. \iff$$

$$\left| \begin{aligned} 2 &= A + 2B \\ 4 &= A + 4B \end{aligned} \right. \iff \left| \begin{aligned} A &= 2 - 2B \\ 4 &= 2 - 2B + 4B \end{aligned} \right. \iff$$

$$\left| \begin{aligned} A &= 2 - 2B \\ B &= 1 \end{aligned} \right. \Rightarrow A = 0$$

Решението на системата е $A = 0$, $B = 1$. Заместваме в 3) и получаваме $k_m = 0 \cdot 2^m + 1 \cdot 4^m = 4^m$. $\square$

Условието за край на цикъла $k_m \le n$ решаваме като неравенство спрямо m и намираме колко пъти се изпълнява:

$$k_m \le n \iff 4^m \le n \iff m \le \log_4 n.$$

Тялото на цикъла се изпълнява $\lfloor \log_4 n \rfloor$ пъти и толкова числа се отпечатват. Това е и времевата сложност на алгоритъма: $\boxed{\Theta(\log n)}$

Задача 3: Разглеждаме следния алгоритъм:

```
ALG_3(A[1...n])
for k ← 2 to n do
    if A[k] | A[k-1]
        return k
return 1
```

Каква стойност връща алгоритъмът ALG_3?

Докажете отговора си с подходящ инвариант и полуинвариант.

Решение:

Алгоритъмът ALG_3 връща най-малкия индекс (от 2 до n вкл.), чийто елемент дели предшественика си. Алгоритъмът връща единица, ако няма такъв елемент.

1. Коректност на изхода

Инвариант: За всяка проверка за край на цикъла елементът A[q] **не дели** A[q-1] за никое q = 2,3,..,k-1.

Доказателство:

База: Първата проверка се изпълнява при влизане в цикъла. Тогава k = 2 и инвариантът е тривиално верен: променливата q с квантор за всеобщност притежава празно множество от допустими стойности. OK!

Поддръжка: Допускаме, че инвариантът е изпълнен за някоя проверка за край на цикъла, **която не е последна**. Тоест A[q] **не дели** A[q-1] за никое q = 2,3,..,k-1.

Щом проверката не е последна, условието A[k] | A[k-1] вътре в тялото на цикъла **връща лъжа**. Но това означава, че **A[k] не дели A[k-1]**.

От допускането знаем, че A[q] **не дели** A[q-1] за никое q = 2,3,..,k-1. Но знаем и, че A[k] **не дели** A[k-1]. Следователно заключаваме, че A[q] **не дели** A[q-1] за никое q = 2,3,..,k.

При следващата проверка за край на цикъла k се увеличава с единица, откъдето получаваме, **че A[q] не дели A[q-1]** за никое q = 2,3,..,k-1. OK!

Завършек: Изпълнението на цикъла може да приключи от две места:

1) От проверката в оператора if (A[k] | A[k-1]). Това означава, че условието A[k] | A[k-1] **връща истина**. Следователно A[k] **дели** A[k-1]. Но от инварианта знаем, че **A[q] не дели A[q-1]** за никое q = 2,3,..,k-1. Следователно елементът на индекс k е първият, който дели предшественика си. Но в тялото на оператора if **връщаме точно k**. OK!

2) Условието за край на цикъла връща лъжа, т.е. **k = n+1**. От инварианта знаем, че **A[q] не дели A[q-1]** за никое q = 2,3,..,k-1. Заместваме k с n+1 и получаваме, че **A[q] не дели A[q-1]** за никое q = 2,3,..,n, т.е. **никой елемент не дели предшественика си**. Веднага след цикъла връщаме точно единица. OK!

2. Алгоритъмът ще приключи

Алгоритъмът ще завърши, защото се състои от единствен цикъл,

който е цикъл по брояч. По-формално, **полуинвариант е броячът k на цикъла**:

стойността на брояча нараства с единица след всяко изпълнение на тялото,

затова от 2 до n + 1 ще достигне за (n + 1) – 2 = **n – 1 изпълнения на тялото**

на цикъла. **Цикълът завършва след най-много n – 1 изпълнения на тялото**,

когато k достигне n + 1 (или по-рано, ако бъде намерен подходящ елемент).