Зад. 1

```
1. Alg1(n):
2.      s←1
3.      for i←1 to n
4.          s←s*2
5.      return s
```

Твърдим, че $\mathrm{Alg1}(n) = 2^n$

Инварианта: При всяко k-то достигане на ред 3 имаме, че $s = 2^{k-1}$ (k броим от 1)

База:

При $k = 1$-во достигане на ред 3

$s \overset{\text{ред 2}}{=} 1 = 2^0 = 2^{1-1} = 2^{k-1}$

Поддръжка:

Нека е вярно за някое k-то достигане на ред 3, което не е последното... т.е имаме, че $s = 2^{k-1}$.

Сега се изпълнява тялото на цикъла.. т.е ред 4 от където имаме $s_{\text{new}} \overset{\text{ред 4}}{=} s_{\text{old}} * 2 = 2^{k-1} * 2 = 2^k$

Терминация:

При $k = (n+1)$-то достигане на ред 3 тялото на цикъла няма да се изпълни.. от поддръжката имаме, че $s = 2^{(n+1)-1} = 2^n$.

Директно след цикъла връщаме $s = 2^n$.

Зад. 2

```
1. Alg2(A[1..n]):
2.      s←0
3.      for i←1 to n
4.          s←s+A[i]
5.      return s
```

Твърдим, че Alg2 (A) събира всички числа от масива A[1..n]

Инварианта: На всяко k-то достигане на ред 3 имаме, че $s = \sum_{j=1}^{k-1} A[j]$

База:

При $k = 1$-во достигане на ред 3

$0 \overset{\text{ред 2}}{=} s = \sum_{j=1}^{0} A[j] = 0$

Поддръжка:

Нека е вярно за някое k-то влизане, което не е последното. Ще докажем, че е изпълнено за (k+1)-то влизане.

От предположението имаме, че $s_{\text{old}} = \sum_{j=1}^{k-1} A[j]$. Сега се изпълнява тялото на цикъла

$s_{\text{new}} \overset{\text{ред 4}}{=} s_{\text{old}} + A[i] \overset{i=k}{=} s_{\text{old}} + A[k] = \sum_{j=1}^{k-1} A[j] + A[k] = \sum_{j=1}^{k+1-1} A[j]$.

Терминация:

От инвариантата знаем, че при достигане на ред 3 за $k = (n+1)$-ви път имаме $s = \sum_{j=1}^{(n+1)-1} A[j]$.

Директно след цикъла връщаме $s = \sum_{j=1}^{n} A[j]$. Т.е програмата ни връща сумата на всички елементи на масива.

Зад. 3

```
1. Alg3(a, n): //a ∈ ℝ
2.      if n=0 then
3.           return 1
4.      if n is even then
5.           return Alg3(a*a, n/2)
6.      return a*Alg3(a, n-1)
```

Note: Тъй като това е рекурсивна програма, то нея ще докажем с познатата ни (пълна) индукция. Разбира се, ако има итеративен цикъл вътре в изпълнението на рекурсивната програма, то ще трябва да правим и инварианта (вътрешно) и индукция (външно).

Твърдим, че $\text{Alg3}(a, n) = \begin{cases} a^n & , a^2 + n^2 \neq 0 \\ 1 & , \text{иначе} \end{cases}$.

За по-лесно може да докажем, че $\text{Alg3}(a, n) = \begin{cases} a^n & , n > 0 \\ 1 & , n = 0 \end{cases}$.

Двете са еквивалентни (не е тривиално ясно, че са еквивалентни разбира се, но проверката за равенство е достатъчно проста).. важното е, че може да докажем кое да от 2-те.

База: $n = 0$

Тогава при изпълнението на Alg3, условието на оператора if на ред 2 ще е TRUE т.е ще се изпълни тялото му и по-конкретно ред 3. Той връща директно 1.

ИП:

Нека $\forall\, m \leq n$ е изпълнено, че $\text{Alg3}(a,\ m) = \begin{cases} a^m & , m > 0 \\ 1 & , m = 0 \end{cases}$

ИС: Ще док., че е вярно за $n + 1$

<u>*сл.1*</u> $(n + 1) \equiv 0 \pmod 2$

Ясно е, че $n + 1 \geq 1$. Тоест няма да влезем в тялото на оператор if от ред 2.

Тогава ще влезем в тялото на оператор if от ред 4 и по-конкретно return $\text{Alg3}(a * a,\ n/2)$.

Имаме $\text{Alg3}(a,\ n + 1) = \text{Alg3}(a * a,\ (n + 1)/2) \overset{\text{ИП}}{=} (a * a)^{(n+1)/2} = \left(a^2\right)^{(n+1)/2} = a^{\frac{2(n+1)}{2}} = a^{n+1}$

<u>*сл.2*</u> $(n + 1) \equiv 1 \pmod 2$

Ясно е, че няма да изпълним нито тялото на оператор if от ред 2, нито тялото на този от ред 4.

Тъй като $\text{Alg3}(a, n)$ има 2 клаузи - $n > 0$ и $n = 0$ трябва да разгледаме 2 подслучая:

<u>*сл.2.1*</u> $n + 1 = 1$

$\text{Alg3}(a,\ 1) = a * \text{Alg3}(a,\ 0) \overset{\text{ИП}}{=} a * \mathbf{1} = a$ //забележете, че пише 1, а не a^0

<u>*сл.2.2*</u> $n + 1 > 1$

$\text{Alg3}(a,\ n + 1) = a * \text{Alg3}(a,\ n + 1 - 1) \overset{\text{ИП}}{=} a * a^n = a^{n+1}$

Зад. 4

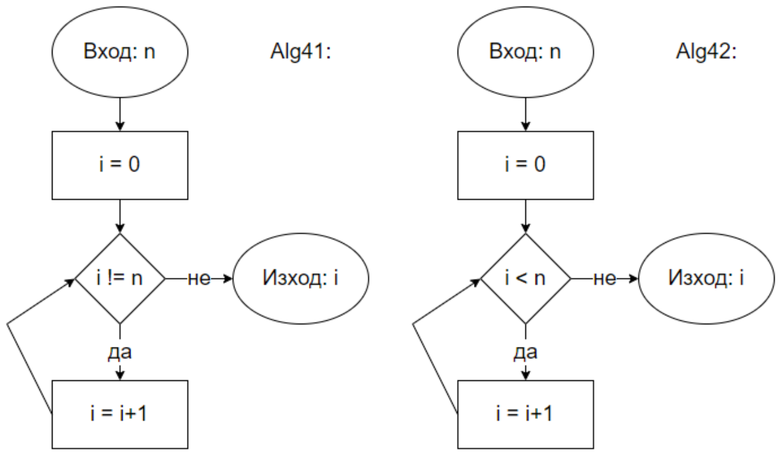
```
1. Alg41(n):
2.      i←0
3.      while i≠n do
4.           i←i+1
5.      return i
```

Инварианта: При всяко k-то достигане на ред 3, имаме че $i = k - 1$.

```
1. Alg42(n):
2.      i←0
3.      while i<n do
4.           i←i+1
5.      return i
```

Инварианта: При всяко k-то достигане на ред 3, имаме че $i = k - 1$ (и още нещо?).

Очевидно (в някакъв смисъл) двата алгоритъма Alg41(n) и Alg42(n) работят по абсолютно аналогичен начин. Единствената им разлика би била, че Alg42(n) не бихме успели да направим терминация както ни се иска. Нека разгледаме блок-схемите на 2-та алгоритъма:



Може да забележим, че ако се озовем в "*Изход: i*" имаме следното:

***a*)** за Alg41(n): $\neg (i \neq n) \equiv i = n$

***b*)** за Alg42(n) : $\neg (i < n) \equiv i \geq n$, от което не можем да кажем нищо за изхода на Alg42(n).. може да върне n, може да върне $n + 1$.. или пък $n + 10^{10^{10}}$. По тази причина, към инвариантата на Alg42(n) трябва да си добавим $i \leq n$. Тоест инвариантата ни ще изглежда по следния начин:

Инварианта: При всяко k-то достигане на ред 3, имаме че $i = k - 1$ и $i \leq n$.

Тогава при достигане на *"Изход: i"* ще имаме следното:

$\neg (i < n) \,\&\, i \leq n \equiv i \geq n \,\&\, i \leq n \equiv i = n$

Може да забелижите, че всъщност $i = k - 1$ ни служи само за това да ни гарантира, че ф-ята ни е тотална. Не ни е нужно за коректността при изхода!

П.С. Ако доказваме с блок схеми, то доказателството минава на 3 отделни части: вход→основна част, основна част→основна част, основна част→изход. Разбира се, ако имаме повече вложени цикли, доказателството ще минава на повече части.

Зад. 5

```
1.  Alg5(A[1..n]) // Kadane's Algorithm
2.      c←A[1]
3.      m←A[1]
4.      for i←2 to n
5.          if A[i]+c > A[i] then
6.              c←c+A[i]
7.          else
8.              c←A[i]
9.          if c > m then
10.             m←c
11.     return m
```

Твърдим, че Alg5($A[1\,..\,n]$) намира най-голята сума на непразен подмасив на масива A[1..n].
Инварианта: На всяко k-то достигане на ред 4 имаме:
- $i = k + 1$
- c е най-голямата сума на непразен подмасив на A[1..k], завършваща в индекс k.
- m е най-голямата сума на непразен подмасив на A[1..k]

База:
На $k = 1$-во влизане имаме:
- $i \overset{\text{ред 4}}{=} 2 = k + 1$
- c изпълнява усл. на инвариантата
- m изпълнява усл. на инвариантата

Поддръжка:
Нека е изпълнено:
- $i_{\text{old}} = k + 1$
- c_{old} е най-голямата сума на непразен подмасив на $A[1\,..\,k]$, завършваща в индекс k.
- m_{old} е най-голямата сума на непразен подмасив на $A[1\,..\,k]$.
 за някое k-то (непоследно) достигане на ред 4.

 Ще докажем за $(k + 1)$-то достигане на ред 4.
- $i_{\text{new}} \overset{\text{ред 4}}{=} i_{\text{old}} + 1 = (k + 1) + 1 = k + 2$

- $c_{\text{new}} \overset{\boxed{1}}{=} \max(c_{\text{old}} + A[i_{\text{old}}], A[i_{\text{old}}])$ //използваме i_{old}, а не i_{new}, защото i се актуализира след изпълнение на тялото на цикъла.
Нека допуснем, че c_{new} не е най-голямата сума на непразен подмасив на $A[1\,..\,k]$, завършваща в индекс $k + 1$. Нека най-голямата такава сума я означим с t. Тоест имаме, че
$t > c_{\text{new}} = \max(c_{\text{old}} + A[i_{\text{old}}], A[i_{\text{old}}]) > c_{\text{old}} + A[i_{\text{old}}]$. Но тогава имаме, че $t - A[i_{\text{old}}] > c_{\text{old}}$, но $i_{\text{old}} = k + 1$..
тоест имаме, че $t - A[k + 1] > c_{\text{old}}$. Но t ни беше максималната сума на непразен подмасив на $A[1\,..\,k + 1]$, завършваща в индекс $k + 1$. Сега като извадим $A[k + 1]$ получаваме, че $t - A[k + 1]$ е най-голямата сума на **(потенциално празен)** подмасив на $A[1\,..\,k]$, завършващ в индекс k.
<u>*сл.1*</u> (подмасива е празен)
    Тогава имаме, че $t = A[k + 1]$.
    Знаем, че $c_{\text{new}} = \max(c_{\text{old}} + A[i_{\text{old}}], A[i_{\text{old}}])$. Сега ще докажем, че $c_{\text{old}} \leq 0$.
    Допускаме противното - допускаме, че $c_{\text{old}} > 0$.
    Тогава имаме, че $c_{\text{new}} = \max(c_{\text{old}} + A[i_{\text{old}}], A[i_{\text{old}}]) \overset{c_{\text{old}} > 0}{=} c_{\text{old}} + A[i_{\text{old}}] \overset{c_{\text{old}} > 0}{>} A[i_{\text{old}}] \overset{i_{\text{old}} = k+1}{=} A[k + 1] = t$.
Противоречие!

Тоест знаем, че $c_{\text{old}} \leq 0$. Тоест $c_{\text{new}} = \max(c_{\text{old}} + A[i_{\text{old}}], A[i_{\text{old}}]) \overset{c_{\text{old}} \leq 0}{=} A[i_{\text{old}}] \overset{i_{\text{old}} = k+1}{=} A[k+1] = t$.

Но бяхме допуснали, че $t > c_{\text{new}} \Rightarrow$ противоречие ($t > c_{\text{new}}$ & $t = c_{\text{new}}$)!

сл.2 (подмасива не е празен)

Тогава имаме, че $t - A[k+1]$ е сума на **непразен** подмасив на $A[1 \,..\, k]$, завършваща в индекс k.

Но тогава от допускането, че c_{old} е най-голямата сума на непразен подмасив на $A[1 \,..\, k]$, завършваща в индекс k и от горното получаваме проиворечие.

Така доказахме, че c_{new} е максимална сума на подмасив на $A[1 \,..\, k+1]$, завършваща в индекс $k+1$.

- $m_{\text{new}} = \max(c_{\text{new}}, m_{\text{old}})$ //използваме c_{new}, а не c_{old}, защото на редове 9-10, където актуализираме m, вече сме актуализирали c.

Ясно е, че m_{new} или ще съдържа $A[k+1]$, или няма. Ако го съдържа, то знаем, че c_{new} е максималния подмасив на $A[1 \,..\, k+1]$, който да го съдържа. Ако не го съдържа, то от предположението знаем, че отговора е в m_{old}. Избираме по-голямото от двете.

Терминация:

За $k = n$ е 1-вия път, в който не се изпълнява тялото на оператора for, т.е се изпълнява ред 11, който ни връща m. От инавиантата имаме, че m ни е максимална сума на непразен подмасив на $A[1 \,..\, n]$ (де факто $A[1 \,..\, k]$, но $n = k$), което трябваше да докажем.

[1] *Доказателството на това твърдение е оставено за читателя.*

Задачи от миналата година, по същата тема:

Зад. 1
Дадена е кутия с 53 сини топки и 42 червени топки. Извън кутията имаме неограничен брой сини и червени топки.

```
1. Alg1():
2.     while "не остане 1 топка в кутията" do
3.         "извади 2 случайни топки от кутията"
4.         if "2-те топки са еднакъв цвят" then
5.             "добави 1 червена"
6.         else
7.             "добави 1 синя"
```

Какъв е цвета на топката, която е останала самичка в кутия в края?

Това както се вижда, не е никакъв конкретен алгоритъм.. та дори не връщаме нищо.. нито печатаме нищо. Въпреки това може да използваме инварианта за да постигнем исканото от нас.

Инварианта: На k-тото достигане на ред 2. имаме: броя топки в кутията е 95-k+1 и имаме нечетен брой сини топки в кутията.