АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ" ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 2. КУРС, 1. ПОТОК (СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 1

Задача 1. Какво връща алгоритъмът ALG_1? Обосновете се с инварианта.

```
ALG_1(A[1...n])
s ← 0
for k ← 1 to n do
   if A[k] < 0
      s ← s - A[k]
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 158
   return
for i ← 1 to n do
   for j ← 1 to n do
      print A[i] + j
p ← n / 3
for k ← 1 to 9 do
   ALG_2(A[k...p])
   p ← p + 1
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 65
   return
for i ← 1 to n do
   print A[i] + 77
p ← n - 1
for k ← 1 to 2 do
   ALG_3(A[k...p])
   p ← p + 1
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ" ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 2. КУРС, 1. ПОТОК (СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 2

Задача 1. Какво връща алгоритъмът ALG_1? Обосновете се с инварианта.

```
ALG_1(A[1...n])
s ← 0
for k ← 1 to n do
   if A[k] > 0
      s ← s + 3
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 54
   return
for i ← 1 to n do
   for j ← i to n do
      print A[j] - i
ALG_2(A[2...n])
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 384
   return
for i ← 1 to n do
   print A[i] + 4
p ← n / 100
for k ← 1 to 10 do
   ALG_3(A[k...p])
   p ← p + 1
```


Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ"
ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 2. КУРС, 1. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 3

Задача 1. Какво връща алгоритъмът ALG_1? Обосновете се с инвариант.

```
ALG_1(A[1...n])
s ← 0
for k ← 2 to n do
   if A[k] > A[k-1]
      s ← s + 1
return s
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 300
   return
for i ← 1 to n do
   for j ← 1 to n do
      print i × A[j]
p ← n / 7
for k ← 1 to 7 do
   ALG_2(A[k...p])
   p ← p + 1
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 25
   return
for i ← 1 to n do
   print A[i] + i
ALG_3(A[1...n-1])
ALG_3(A[2...n-1])
ALG_3(A[1...n-2])
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

АНАЛИЗ НА АЛГОРИТМИ
ПРИМЕРНО КОНТРОЛНО № 1 ПО "ДИЗАЙН И АНАЛИЗ НА АЛГОРИТМИ"
ЗА СПЕЦИАЛНОСТ "КОМПЮТЪРНИ НАУКИ", 2. КУРС, 1. ПОТОК
(СУ, ФМИ, ЛЕТЕН СЕМЕСТЪР НА 2019 / 2020 УЧ. Г.)

ВАРИАНТ № 4

Задача 1. Какво прави алгоритъмът ALG_1? Обосновете се с инвариант.

```
ALG_1(A[1...n])
for k ← 1 to n do
   A[k] ← -A[k]
```

Задача 2. Намерете времевата сложност на алгоритъма ALG_2.

```
ALG_2(A[1...n])
if n < 46
   return
for i ← 1 to n do
   s ← 1
   p ← 1
   while p ≤ n do
      s ← s + 1
      p ← s × s
      print A[p - s]
ALG_2(A[1...n-1])
```

Задача 3. Намерете времевата сложност на алгоритъма ALG_3.

```
ALG_3(A[1...n])
if n < 678
   return
for i ← 1 to n do
   print A[i] + 71
p ← n / 10
for k ← 1 to 100 do
   ALG_3(A[k...p])
   p ← p + 1
```

Задача 4. Сравнете по времева сложност алгоритмите ALG_2 и ALG_3.

РЕШЕНИЯ

ВАРИАНТ № 1

Задача 1.

Алгоритъмът `ALG_1` връща сбора от абсолютните стойности на отрицателните числа в дадения масив.

Инварианта:

Стойността на `s` е сборът от абсолютните стойности на отрицателните числа в подмасива `A[1...k-1]`.

Задача 2.

Рекурентно уравнение:
$T(n) = 9T(n/3) + \Theta(n^2)$.

От втория случай на мастър-теоремата намираме решението:
$T(n) = \Theta(n^2 \log n)$.

Задача 3.

Рекурентно уравнение:
$T(n) = 2T(n-1) + \Theta(n)$.

С характеристично уравнение намираме решението:
$T(n) = \Theta(2^n)$.

Задача 4. Алгоритъмът `ALG_2` е по-бърз от `ALG_3`. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

ВАРИАНТ № 2

Задача 1.

Алгоритъмът `ALG_1` връща утроения брой на положителните числа в дадения масив.

Инварианта:

Стойността на `s` е равна на утроения брой на положителните числа в подмасива `A[1...k-1]`.

Задача 2.

Рекурентно уравнение:
$T(n) = T(n-1) + \Theta(n^2)$.

С характеристично уравнение или с помощта на развиване намираме решението:
$T(n) = \Theta(n^3)$.

Задача 3.

Рекурентно уравнение:
$T(n) = 10T(n/100) + \Theta(n)$.

От третия случай на мастър-теоремата намираме решението:
$T(n) = \Theta(n)$.

Задача 4. Алгоритъмът `ALG_3` е по-бърз от `ALG_2`. Сложностите им се сравняват, като се пресметне границата на тяхното частно. Логаритмуването не е приложимо тук.

ВАРИАНТ № 3

Задача 1.

Алгоритъмът `ALG_1` връща броя на елементите, които са по-големи от предходния елемент.

Инварианта:

`s` = броя на елементите в подмасива `A[2...k-1]`, които са по-големи от предходния елемент.

Задача 2.

Рекурентно уравнение:
$T(n) = 7T(n/7) + \Theta(n^2)$.

От третия случай на мастър-теоремата намираме решението:
$T(n) = \Theta(n^2)$.

Задача 3.

Рекурентно уравнение:
$T(n) = T(n-1) + 2T(n-2) + \Theta(n)$.

С характеристично уравнение намираме решението:
$T(n) = \Theta(2^n)$.

Задача 4. Алгоритъмът `ALG_2` е по-бърз от `ALG_3`. Сравняването на сложностите им се извършва най-лесно чрез логаритмуване.

ВАРИАНТ № 4

Задача 1.

Алгоритъмът `ALG_1`
сменя знаците на числата
в дадения масив `A[1...n]`.

Инвариантa:

Алгоритъмът `ALG_1`
е сменил знаците на числата
в подмасива `A[1...k-1]`.

Задача 2.

Рекурентно уравнение:
$T(n) = T(n-1) + \Theta(n^{3/2})$.

С помощта на развиване
намираме решението:
$T(n) = \Theta(n^{5/2})$.

Тук не може да се използва
характеристично уравнение,
тъй като свободният член
е от дробна степен.

Задача 3.

Рекурентно уравнение:
$T(n) = 100\,T(n/10) + \Theta(n)$.

От първия случай
на мастър-теоремата
намираме решението:
$T(n) = \Theta(n^2)$.

Задача 4. Алгоритъмът `ALG_3` е по-бърз от `ALG_2`. Сложностите им се сравняват, като се пресметне границата на тяхното частно. Логаритмуването не е приложимо тук.

СХЕМА ЗА ТОЧКУВАНЕ

Всяка задача носи по 5 точки, а цялото контролно — най-много 20 точки.

Задача 1. Дава се по една точка за всяка от следните стъпки:
— формулиране на вярна и използваема инвариантa;
— доказателство на базата на инвариантата;
— индуктивна стъпка (поддръжка на инвариантата);
— доказателство, че алгоритъмът ще завърши;
— извод за върнатата стойност.
Ако не е формулирана вярна и използваема инвариантa, не се дават точки по тази задача.

Задача 2 и задача 3. Петте точки се разпределят по следния начин:
— за съставяне на рекурентно уравнение: 2 точки;
— за решаване на рекурентното уравнение: 3 точки.
Ако уравнението е грешно съставено, не се дават точки, независимо дали и как е решено.

Задача 4. Ако асимптотичното сравнение се прави чрез граници,
се дават 4 точки за пресмятане на границата и 1 точка за отговора.
Ако асимптотичното сравнение се извършва чрез логаритмуване,
се дават 2 точки за логаритмуването, 2 точки за границата и 1 точка за отговора.

Тези правила се прилагат, ако е избран най-рационалният начин за решаване на задачата. В противен случай се отнема 1 точка.

Тъй като отговорът може да бъде предположен с помощта на асимптотичния ред, то при грешен отговор не се дават никакви точки за другите етапи от решението.

Ако отговорът е верен, но недостатъчно обоснован, не се дават точки за отговора, а само за преодолените етапи от обосновката.

Точки по задача 4 се дават само ако са правилно пресметнати времевите сложности на алгоритмите `ALG_2` и `ALG_3`. За сравняване на една или две погрешни сложности не се дават точки.