Критерии оценивания заданий с развёрнутым ответом
Вариант 1

24 На обработку поступает последовательность из четырёх неотрицательных целых чисел (некоторые числа могут быть одинаковыми). Нужно написать программу, которая выводит на экран количество кратных 6 чисел в исходной последовательности и минимальное кратное 6 число. Если чисел, кратных 6, нет, требуется на экран вывести «NO». Известно, что вводимые числа не превышают 1000. Программист написал программу неправильно. Ниже эта программа для Вашего удобства приведена на пяти языках программирования.

Напоминание: 0 делится на любое натуральное число.

Бейсик	**Python**

CONST n = 4
count = 0
minimum = 0
FOR I = 1 TO n
  INPUT x
  IF x mod 6 = 0 THEN
  count = count + 1
  IF x > minimum THEN
  minimum = x
  END IF
  END IF
NEXT I
IF count> 0 THEN
  PRINT count
  PRINT minimum
ELSE
  PRINT "NO"
END IF
	n = 4
	count = 0
	minimum = 0
	for i in range(1, n+1):
	  x = int(input())
	  if x % 6 == 0:
	  count += 1
	  if x > minimum:
	  minimum = x
	if count > 0:
	  print(count)
	  print(minimum)
	else:
	print("NO")

Алгоритмический язык

```
алг
нач
цел n = 4
целi, x, minimum, count
  count := 0
  minimum := 0
нцдляi от 1 доn
вводx
еслиmod(x, 6) = 0 то
count := count + 1
если x > minimum то
        minimum:= x
все
все
кц
еслиcount> 0 то
выводcount, нс
вывод minimum
иначе
вывод "NO"
все
кон
```

Паскаль

```
const n = 4;
var i, x, minimum, count: integer;
begin
  count := 0;
  minimum := 0;
  for i := 1 to n do
  begin
    read(x);
    if x mod 6 = 0 then
    begin
      count := count + 1;
      if x > minimum then
        minimum := x
    end
  end;
  if count > 0 then
  begin
    writeln(count);
    writeln(minimum);
  end
  else
    writeln('NO')
end.
```

Си

```
#include <stdio.h>
#define n 4

int main()
{
  int i, x, minimum, count;
  count = 0;
  minimum = 0;
  for (i = 1; i <= n; i++)
  {
    scanf("%d",&x);
    if (x % 6 == 0)
    {
      count++;
      if (x > minimum)
        minimum = x;
    }
  }
  if (count > 0)
  {
    printf("%d\n", count);
    printf("%d\n", minimum);
  }
  else
    printf("NO\n");
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе последовательности:
24, 21, 18, 15.

2. Приведите пример последовательности, в которой есть хотя бы одно кратное 6 число, при вводе которой, несмотря на ошибки, программа печатает правильный ответ.

3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:

   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Достаточно указать ошибки и способ их исправления для одного языка программирования.

Обратите внимание, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения. Исправление ошибки должно затрагивать только строку, в которой находится ошибка.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Решение использует запись программы на Паскале. Допускается использование записи программы на любом из четырёх других языков программирования.

1. Программа выведет два числа: 2 и 24.

2. Пример последовательности, содержащей кратные 6 числа, для которой программа выдаёт правильный ответ: 16, 26, 36, 46.

Замечание для проверяющего. В конце работы программы значение переменной minimum всегда равно максимальному кратному 6 числу или 0, если в последовательности нет чисел, кратных 6. Соответственно, программа будет работать верно, если в последовательности максимальное кратное 6 число равно минимальному, т.е. в последовательности должно быть одно число, кратное 6, или несколько одинаковых чисел, кратных 6. Выведенное количество кратных 6 чисел будет правильным в любом случае.

3. В программе есть две ошибки.

Первая ошибка. Неверная инициализация ответа (переменная `minimum`).

Строка с ошибкой:

```
minimum := 0;
```

Верное исправление:

```
minimum := 1001;
```

Вместо 1001 может быть любое целое число, большее 996, либо MAXINT.

Можно использовать и число 996, так как при выводе мы проверяем, есть ли в последовательности хотя бы одно кратное 6 число.

Вторая ошибка. Неверное условие перевычисления минимума.
Строка с ошибкой:
```
if x >minimum then
```
Верное исправление:
```
if x <minimum then
```

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **четыре** действия: 1) указать, что выведет программа при конкретном входном числе; 2) указать пример входной последовательности, при вводе которой программа выдаёт правильный ответ; 3) исправить первую ошибку; 4) исправить вторую ошибку. Для проверки правильности выполнения п. 2) нужно формально выполнить исходную (ошибочную) программу с входными данными, которые указал экзаменуемый, и убедиться в том, что результат, выданный программой, будет таким же, как и для правильной программы. Для действий 3) и 4) ошибка считается исправленной, если выполнены оба следующих условия: а) правильно указана строка с ошибкой; б) указан такой новый вариант строки, что при исправлении другой ошибки получается правильная программа	
Выполнены все четыре необходимых действия, и ни одна верная строка не указана в качестве ошибочной	3
Не выполнены условия, позволяющие поставить 3 балла. Имеет место одна из следующих ситуаций: а) выполнены три из четырёх необходимых действий. Ни одна верная строка не указана в качестве ошибочной; б) выполнены все четыре необходимых действия. Указано в качестве ошибочной не более одной верной строки	2
Не выполнены условия, позволяющие поставить 2 или 3 балла. Выполнены два необходимых действия из четырёх	1
Не выполнены условия, позволяющие поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

25 Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от –10 000 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, позволяющий найти и вывести количество таких троек элементов массива, в которых средний элемент равен произведению двух крайних элементов тройки. В данной задаче под тройкой подразумевается три подряд идущих элемента массива.
Например, для массива из пяти элементов: 2; 4; 2; 3; 1 – ответ: 1.

Исходные данные объявлены так, как показано ниже на примерах для пяти языков программирования. Запрещается использовать переменные, не описанные ниже, но разрешается не использовать некоторые из описанных переменных.

Бейсик

```
CONST N AS INTEGER = 20
DIM A (1 TO N) AS LONG
DIM I AS LONG,
 J AS LONG,
 K AS LONG

FOR I = 1 TO N
    INPUT A(I)
NEXT I
...

END
```

Python

```
# допускается также
# использовать две
# целочисленные переменные j и k
a = []
n = 20
for i in range(0, n):
    a.append(int(input()))
...
```

Алгоритмический язык

```
алг
нач
целN = 20
целтабa[1:N]
целi, j, k
нцдляiот 1 доN
ввод a[i]
кц
    ...

кон
```

Паскаль

```
const
    N = 20;
var
a: array [1..N] of longint;
    i, j, k: longint;
begin
    for i := 1 to N do
        readln(a[i]);
    ...

end.
```

Си

```
#include <stdio.h>
#define N 20
int main() {
long a[N];
long i, j, k;
    for (i = 0; i<N; i++)
        scanf("%ld", &a[i]);
    ...
    return 0;
}
```

В качестве ответа Вам необходимо привести фрагмент программы (или описание алгоритма на естественном языке), который должен находиться на месте многоточия. Вы можете записать решение также на другом языке программирования (укажите название и используемую версию языка программирования, например Free Pascal 2.6) или в виде блок-схемы. В этом случае Вы должны использовать те же самые исходные данные и переменные, какие были предложены в условии (например, в образце, записанном на Алгоритмическом языке).

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

На языке Паскаль

```
k := 0;
for i := 1 to N-2 do
    if (a[i+1]=a[i]*a[i+2]) then
        inc(k);
writeln(k);
```

На алгоритмическом языке

```
k := 0;
нцдля i от 1 до N-2
если a[i+1]=a[i]*a[i+2]
то
k := k+1
все
кц
выводk
```

На языке Бейсик

```
K = 0
FOR I = 2 TO N-1
    IF A(I)=A(I-1)*A(I+1) THEN
        K = K+1
    END IF
NEXT I
PRINT K
```

На языке Си

```
k = 0;
for (i = 1; i<N-1; i++)
    if (a[i]==a[i-1]*a[i+1])
        k++;
printf("%ld", k);
```

На языке Python	

`k = 0`
`for i in range(1, n – 1):`
`    if (a[i]==a[i-1]*a[i+1]):`
`        k += 1`
`print(k)`	

Указания по оцениванию	**Баллы**
Общие указания. 1. В программе допускается наличие отдельных синтаксических ошибок, не искажающих замысла её автора. 2. Эффективность алгоритма не имеет значения и не оценивается. 3. Допускается запись алгоритма на языке программирования, отличном от языков, приведённых в условии. В этом случае должны использоваться переменные, аналогичные описанным в условии. Если язык программирования использует типизированные переменные, описания переменных должны быть аналогичны описаниям переменных на Алгоритмическом языке. Использование нетипизированных или необъявленных переменных возможно только в случае, если это допускается языком программирования; при этом количество переменных и их идентификаторы должны соответствовать условию задачи	
Предложен правильный алгоритм, выдающий в качестве результата верное значение	2

Не выполнены условия, позволяющие поставить 2 балла. Предложено в целом верное решение, содержащее не более одной ошибки из числа следующих: 1) в цикле происходит выход за границу массива слева или справа (например, используется цикл от 1 до *N*); 2) не инициализируется или неверно инициализируется счётчик количества найденных троек; 3) счётчик количества троек в цикле не изменяется или изменяется неверно; 4) неверно указаны индексы элементов тройки при сравнении произведения крайних элементов со средним; 5) средний элемент сравнивается не с произведением, а с суммой крайних элементов; 6) используется неправильный знак сравнения; 7) отсутствует вывод ответа; 8) используется переменная, не объявленная в разделе описания переменных; 9) не указано или неверно указано условие завершения цикла; 10) индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно; 11) неверно расставлены операторные скобки	1
Ошибок, перечисленных в п. 1–11, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии цикла в явном или неявном виде)	0
Максимальный балл	*2*

26 Два игрока, Петя и Ваня, играют в следующую игру. Дан набор слов, составленных из букв русского алфавита, при этом ни одно из заданных слов не является началом другого. Слова в этой игре – это просто цепочки букв, они не обязаны быть осмысленными словами русского языка. Игра состоит в том, что игроки составляют слово из набора, приписывая по очереди буквы к концу составляемого слова, т.е. справа. При этом каждое промежуточное слово должно быть началом одного из заданных слов. Выигрывает тот, кто получит одно из заданных слов целиком. Первый ход делает Петя, т.е. Петя пишет первую букву составляемого слова.

Пример. Заданный набор слов: {АНТАРКТИДА, АНТРАЦИТ, АБАРА, АБАЖУР, ББВ, БАОБАБ, БАР}.

Первым ходом Петя пишет Б (он мог написать Б или А).

Ваня в ответ дописывает А и получает БА (он мог ещё получить ББ).

Вторым ходом Петя получает БАР и выигрывает.

В заданиях используются следующие понятия. *Стратегия* игрока – это правило, указывающее игроку ход, который он должен сделать. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника. Стратегия игрока называется *выигрышной*, если игрок выигрывает в любой партии, разыгранной в соответствии с этой стратегией, как бы ни играл противник.

В описание выигрышной стратегии **не следует** включать ходы играющего по этой стратегии игрока, не являющиеся для него безусловно выигрышными, т.е. не являющиеся выигрышными независимо от игры противника.

Задание 1. а) Укажите, у кого есть выигрышная стратегия при исходном наборе слов {АББВГДАБВГДХ,ДГВББАДГВББА}. Ответ обоснуйте.

б) Укажите, у кого есть выигрышная стратегия при исходном наборе слов {ТРИТРИ…ТРИ, РИТАРИТА…РИТА} (в первом слове ТРИ повторено 33 раза, т.е. его длина 99 букв; во втором слове РИТА повторено 44 раза, т.е. его длина 176 букв). Ответ обоснуйте.

Задание 2. В задании 1а поменяйте местами две буквы в более коротком слове так, чтобы теперь выигрышная стратегия была у другого игрока. Напишите полученный набор слов.

Задание 3. Рассмотрим набор слов {ДРАТВА, ДРОН, ДРОЗД, КРОНА, КРОШКА, КРОКОДИЛИЩЕ}. У кого из игроков есть выигрышная стратегия для этого набора? Приведите в виде рисунка или таблицы дерево всех партий, возможных при этой стратегии. Дерево не должно содержать партий, невозможных при реализации выигрывающим игроком своей выигрышной стратегии. Например, полное дерево игры не является верным ответом на это задание.

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки ответа, не искажающие его смысла)

Примечание для проверяющего. Если в наборе только одно слово, то все ходы игроков единственно возможные. Если длина слова нечётная, то выигрывает Петя; если длина чётная – Ваня. Аналогично, пусть для некоторого слова существует единственная возможность продолжить его до слова из набора. Если длина продолжения нечётная, то выигрывает тот игрок, который ходит; если длина чётная, выигрывает его противник.

Задание 1. а) Выигрышная стратегия есть у Пети. Первым ходом он пишет А. В конце игры у игроков должно получиться слово АБВГДАБВГДХ. Игрокам осталось написать 10 букв, ходить нужно Ване. Поэтому последнюю букву напишет Петя.
б) Выигрышная стратегия есть у Пети. Первым ходом он пишет Т. После этого у игроков должно получиться слово ТРИТРИ…ТРИ. В этом слове 99 букв. Петя напишет 1-ю, 3-ю, 5-ю и т.д. буквы. Поэтому 99-ю букву напишет Петя, и, следовательно, выиграет Петя.

Задание 2. Нужно в слове ДГВБАДГВБАпоставить на первое место букву А. Например, можно поменять местами первую и последнюю буквы; получится слово АГВБАДГВБД. Теперь оба слова начинаются с буквы А, и Петя должен написать А. Ваня пишет Г, получается АГ. Теперь в конце игры получится слово АГВБАДГВБД; до этого осталось написать 8 букв, первую из них будет писать Петя. Поэтому выиграет Ваня.
Примечание для проверяющего. Можно поменять первую букву в слове ДГВБАДГВБА с буквой А в середине слова, это тоже приводит к верному решению.

Задание 3. Выигрышная стратегия есть у Вани. Дерево всех партий для этой стратегии показано в таблице 1 и на рисунке 1. *Примечание для проверяющего*. Оба способа изображения допустимы. Ученику достаточно привести один из них.

Начальная позиция	1-й ход Пети (разобраны все ходы)	1-й ход Вани (только ход по стратегии)	2-й ход Пети (разобраны все ходы)	2-й ход Вани (только ход по стратегии)	3-й ход Пети (разобраны все ходы)	3-й ход Вани (только ход по стратегии)
[пусто]	**Д**	**ДР**	**ДРА**	**ДРАТ**	**ДРАТВ**	**ДРАТВА**
			ДРО	**ДРОН**		
	К	**КР**	**КРО**	**КРОШ**	**КРОШК**	**КРОШКА**

Таб. 1. Дерево всех партий при описанной стратегии Вани. Подчёркнуты

позиции, в которых партии заканчиваются. Указаны только позиции, полученные после ходов игроков. Сам ход – это последняя буква полученного слова. *Примечание для проверяющего.* Ученик может не писать такого подробного комментария.

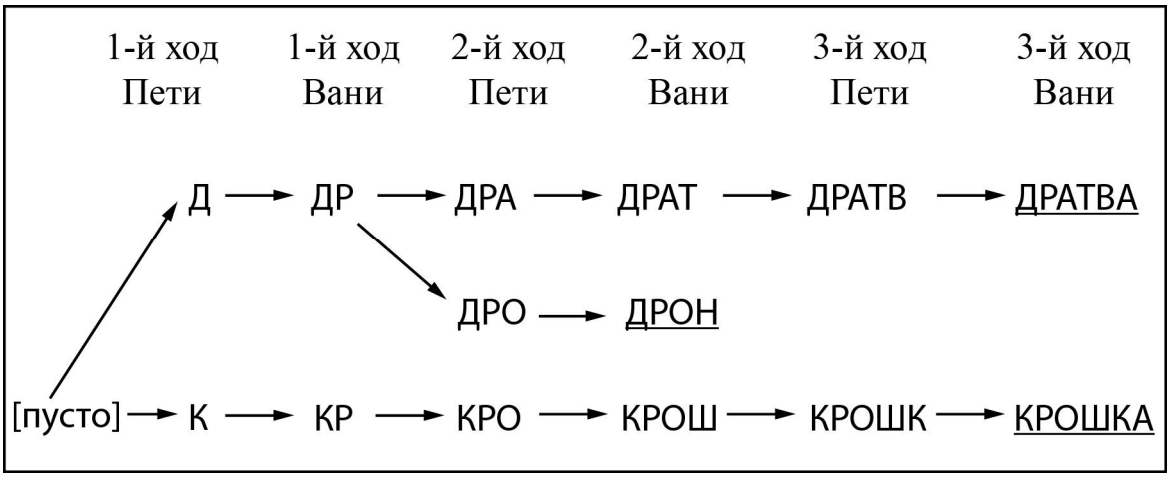

Рис. 1. Дерево всех партий при описанной стратегии Вани. Подчёркнуты позиции, в которых партии заканчиваются. Указаны только позиции, полученные после ходов игроков. Сам ход – это последняя буква полученного слова. *Примечание для проверяющего.* Ученик может не писать такого подробного комментария.

Указания по оцениванию	**Баллы**
Предварительные замечания В задаче от ученика требуется выполнить три задания. Их трудность возрастает. Количество баллов в целом соответствует количеству выполненных заданий (подробнее см. ниже). Описка в решении, не искажающая основного замысла и не приведшая к неверному ответу, при оценке решения не учитывается. ***Задание 1 считается выполненным***, если выполнены оба пункта: 1а и 1б. Пункт 1а считается выполненным, если • верно указан первый ход Пети; • приведена последовательность всех остальных ходов Пети **или** указано, что Пети выигрывает поскольку длина слова АБВГДАБВГДХ нечетна. Пункт 1б считается выполненным, если • верно указан первый ход Пети; • приведена последовательность всех остальных ходов Пети **или** указано, что Пети выигрывает поскольку длина слова ТРИТРИ…ТРИ нечетна. ***Задание 2 считается выполненным,*** если верно указано какие буквы нужно поменять местами, и приведен первый ход	

выигрывающего игрока. ***Задание 3 считается выполненным***, если правильно построено дерево всех возможных партий для выигрышной стратегии Вани. Если в дереве указаны пути, которые не могут встретиться при использовании выигрышной стратегии, то задание считается не выполненным.	
Выполнены задания 1, 2 и 3.	3
Не выполнены условия, позволяющие поставить 3 балла, и выполнено одно из следующих условий. 1. Выполнено задание 3. 2. Выполнены задания 1 и 2.	2
Не выполнены условия, позволяющие поставить 3 или 2 балла, и выполнено одно из следующих условий. 1. Выполнено задание 1. 2. Выполнено задание 2.	1
Не выполнено ни одно из условий, позволяющих поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

27 На вход программы поступает последовательность из N целых положительных чисел, все числа в последовательности различны. Рассматриваются все пары различных элементов последовательности (элементы пары не обязаны стоять в последовательности рядом, порядок элементов в паре не важен). Необходимо определить количество пар, для которых произведение элементов делится на 57.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел N ($1 \le N \le 1000$). В каждой из последующих N строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна напечатать одно число: количество пар, в которых произведение элементов кратно 57.

Пример входных данных:

```
4
3
6
19
38
```

Пример выходных данных для приведённого выше примера входных данных:

```
4
```

Пояснение. Из четырёх заданных чисел можно составить 6 попарных произведений: 3·6, 3·19, 3·38, 6·19, 6·38, 19·38 (результаты: 18, 57, 114, 114, 228, 722). Из них на 57 делятся 4 произведения (3·19=57; 3·38=114; 6·19=114; 6·38=228).

Требуется написать эффективную по времени и по памяти программу для решения описанной задачи.

Программа считается эффективной по времени, если при увеличении количества исходных чисел N в k раз время работы программы увеличивается не более чем в k раз.

Программа считается эффективной по памяти, если память, необходимая для хранения всех переменных программы, не превышает 1 килобайта и не увеличивается с ростом N.

Максимальная оценка за правильную (не содержащую синтаксических ошибок и дающую правильный ответ при любых допустимых входных данных) программу, эффективную по времени и по памяти, – 4 балла.

Максимальная оценка за правильную программу, эффективную только по времени – 3 балла.

Максимальная оценка за правильную программу, не удовлетворяющую требованиям эффективности, – 2 балла.

Вы можете сдать **одну** программу или **две** программы решения задачи (например, одна из программ может быть менее эффективна). Если Вы сдадите две программы, то каждая из них будет оцениваться независимо от другой, итоговой станет **бóльшая** из двух оценок.

Перед текстом программы обязательно кратко опишите алгоритм решения. Укажите использованный язык программирования и его версию.

Содержание верного ответа (допускаются иные формулировки ответа, не искажающие его смысла)
Произведение двух чисел делится на 57, если выполнено одно из следующих условий (условия не могут выполняться одновременно). А. Оба сомножителя делятся на 57. Б. Один из сомножителей делится на 57, а другой не делится. В. Ни один из сомножителей не делится на 57, но один сомножитель делится на 3, а другой – на 19. *Примечание для проверяющего*. Условие делимости произведения на 57 можно сформулировать проще, например, так: (один из сомножителей делится на 57) ИЛИ (один сомножитель делится на 3, а другой – на 19). Но в этом случае пара сомножителей может удовлетворять обоим условиям, что затруднит подсчёт количества пар. При вводе чисел можно определять, делится ли каждое из них на 57, 3 и 19, и подсчитывать следующие значения: 1) *n57* – количество чисел, кратных 57; 2) *n19* – количество чисел, кратных 19, но не кратных 57; 3) *n3* – количество чисел, кратных 3, но не кратных 57. *Примечание для проверяющего*. Сами числа при этом можно не хранить. Каждое число учитывается не более чем в одном из счётчиков. Количество пар, удовлетворяющих условию А, можно вычислить по формуле $n57\cdot(n57 - 1)/2$. Количество пар, удовлетворяющих условию Б, можно вычислить по формуле $n57\cdot(N - n57)$. Количество пар, удовлетворяющих условию В, можно вычислить по формуле $n3\cdot n19$. Поэтому искомое количество пар вычисляется по формуле $n57\cdot(n57 - 1)/2 + n57\cdot(N - n57) + n3\cdot n19$. Ниже приведена реализующая описанный алгоритм программа на языке Паскаль (использована версия PascalABC)

Пример 1. Программа на языке Паскаль. Программа эффективна по времени и по памяти

```
var
    N: integer;     {количество чисел}
a: integer;     {очередное число}
n57, n19, n3: integer;
k57: integer;    {количество требуемых пар}
i: integer;

begin
  readln(N);
  n57:=0; n19:=0; n3:=0;
  for i:=1 to N do begin
    readln(a);
    if a mod 57 = 0 then
      n57 := n57+1
    else if a mod 19 = 0 then
      n19 := n19 + 1
    else if a mod 3 = 0 then
n3 := n3 + 1;
  end;
  k57 := n57*(n57-1) div 2 + n57*(N-n57) + n3*n19;
  writeln(k57)
end.
```

Комментарии для проверяющего

1. При таком решении каждое прочитанное число обрабатывается (делаются проверки делимости, изменяются счётчики) и после этого не хранится. Таким образом, используемая память не зависит от длины последовательности. Время обработки очередного числа фиксировано, т.е. не зависит от длины последовательности. Время заключительных вычислений по приведённой в решении формуле также не зависит от длины последовательности. Поэтому при увеличении длины последовательности в *k*раз время работы программы увеличивается не более чем в*k*раз. Таким образом, приведённая выше программа эффективна как по времени, так и по используемой памяти. Это решение оценивается 4 баллами.

2. Общая идея решения, эффективного по времени, состоит в следующем. Просматриваем по очереди все элементы последовательности и накапливаем значения вспомогательных величин (в приведённом решении это счётчики$n3$, $n19$, $n57$). После того как вся последовательность обработана и подсчитаны окончательные значения вспомогательных величин, по этим значениям подсчитывается искомое количество пар.

При этом можно использовать и другие вспомогательные величины. Например, можно вместо $n3$ и $n19$ использовать величины $p3$ и $p19$ – количества чисел, которые делятся соответственно на 3 и на 19. Так как $n3 = p3 - n57$ и $n19 = p19 - n57$, то итоговая формула примет вид:

$$n57\cdot(n57 - 1)/2 + n57\cdot(N - n57) + (p3 - n57)\cdot(p19 - n57).$$

Ещё один возможный вариант (есть и другие!) – подсчёт количества чисел,

которые не делятся на 57, – можно вести по формуле $n3+n19+nx$, где nx – количество чисел, которые не делятся ни на 3, ни на 19. Значение nx можно вычислить с помощью отдельного счётчика. Такая программа на языке Бейсик приведена ниже.

Все подобные программы оцениваются в 4 балла.

При любом наборе вспомогательных величин возможны различные способы записи итоговой формулы. Можно, например, раскрывать скобки и приводить подобные члены или, наоборот, выносить за скобки общие множители; можно вводить дополнительные переменные для отдельных слагаемых, а затем вычислять их сумму. Допустим любой способ записи вычислений, эквивалентный правильной формуле, выбранный способ записи не влияет на оценку.

3. Возможно решение, основанное на описанных идеях, однако предварительно сохраняющее элементы последовательности в массив. Такое решение (если в нём нет ошибок) эффективно по времени, но неэффективно по памяти. Оно оценивается в 3 балла.

4. Решение, не эффективное ни по времени, ни по памяти, запоминает входную последовательность в массиве, после чего явно перебирает все возможные пары. Такое решение оценивается в 2 балла (см. критерии)

Пример 2. Программа на языке Бейсик. Программа эффективна по времени и по памяти, но использует формулы, отличные от формул программы из примера 1

```
N57 = 0
N3 = 0
N19 = 0
NX = 0
INPUT N
FOR I = 1 TO N
  INPUT A
IF A MOD 57 = 0 THEN
    N57 = N57 + 1
  ELSE
    IF A MOD 19 = 0 THEN
      N19 = N19 + 1
    ELSE
      IF A MOD 3 = 0 THEN
        N3 = N3 + 1
      ELSE NX = NX + 1
      END IF
    END IF
  END IF
NEXT I
K57 = N57*(N57-1)\2 + N57*(N3+N19+NX) + N3*N19
PRINT K57
```

Указания по оцениванию	Баллы
Если в работе представлены две программы решения задачи, то каждая из них независимо оценивается по указанным ниже критериям, итоговой считается бóльшая из двух оценок. Описание алгоритма решения не оценивается.	
Программа правильно работает для любых входных данных произвольного размера. Используемая память не зависит от количества прочитанных чисел, а время работы пропорционально этому количеству. Допускается наличие в тексте программы до трёх синтаксических ошибок одного из следующих видов: 1) пропущен или неверно указан знак пунктуации; 2) неверно написано или пропущено зарезервированное слово языка программирования; 3) не описана или неверно описана переменная; 4) применяется операция, не допустимая для соответствующего типа данных. Если одна и та же ошибка встречается несколько раз, это считается за одну ошибку	4
Не выполнены условия, позволяющие поставить 4 балла. Время работы программы пропорционально количеству введённых чисел; правильно указано, какие величины должны вычисляться по ходу чтения элементов последовательности чисел. Количество синтаксических ошибок («описок») указанных в критериях на 4 балла, не более пяти. Допускается наличие не более **одной** ошибки, принадлежащей к одному из следующих видов: 1) допущена ошибка при вводе данных, например не считывается значение N, или числа могут быть считаны, только если будут записаны в одной строке через пробел; 2) ошибка при инициализации или отсутствие инициализации счётчиков; 3) в программе перепутаны знак целочисленного деления и взятия остатка или знаки операций «равно» и «не равно», „or" вместо „and" и т.п.; 4) использована неверная структура проверок, в результате которой некоторые счётчики могут получить неверное значение; 5) получены правильные значения счётчиков (вспомогательных величин), которые в принципе позволяют получить требуемое количество пар, но формула для вычисления записана неверно (комбинаторная ошибка); 6) отсутствует вывод ответа, или выводится значение не той переменной;	3

7) в описании алгоритма правильно описан смысл используемых вспомогательных величин, и в программе правильно записан алгоритм вычисления искомого количества пар, исходя из этих величин, однако при вычислении одной из вспомогательных величин допущена ошибка; 8) использована одна переменная (или константа) вместо другой; 9) используется один знак операции вместо другого; 10) используется одно зарезервированное слово языка программирования вместо другого. 3 балла также ставится за программу, в которой нет содержательных ошибок, но используемая память зависит от количества прочитанных чисел (например, входные данные запоминаются в массиве, контейнере STL в C++ или другой аналогичной структуре данных)	

Не выполнены условия, позволяющие поставить 3 или 4 балла.
Программа работает в целом верно, эффективно или нет.
В реализации алгоритма допускается до трёх содержательных ошибок, описанных в критериях на 3 балла.
Количество синтаксических ошибок, указанных в критериях на 4 балла, не должно быть более семи.
2 балла также ставится за корректное переборное решение, в котором все числа сохраняются в массиве (или другой аналогичной структуре), рассматриваются все возможные пары и подсчитывается количество подходящих произведений. Пример фрагмента соответствующей программы на языке Паскаль:
`k := 0;`
`fori := 1 ton - 1 do`
`for j := i + 1 to n do`
`  if a[i]*a[j] mod 57 = 0 then`
`  k := k + 1;`
`writeln(k);`
В реализации переборного алгоритма не допускаются логические ошибки, например когда учитываются произведения вида `a[i]*a[i]` или пары считаются дважды
	2

Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. Из текста программы видно, что экзаменуемый в целом правильно представляет путь решения задачи.	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл	*4*