Критерии оценивания заданий с развёрнутым ответом

24 Дано целое положительное число N, не превосходящее 1000. Нужно написать программу, которая определяет, является ли это число степенью числа 4: выводит на экран либо такое целое число K, что $4^K = N$, либо сообщение «NO», если такого числа не существует.

Программист написал программу неправильно. Ниже эта написанная им программа для Вашего удобства приведена на пяти языках программирования.

Бейсик

```
DIM N, K AS INTEGER
INPUT N
K = 0
WHILE N MOD 4 = 0
  K = K + N \ 4
  N = N \ 4
WEND
IF N > 0 THEN
  PRINT K
ELSE
  PRINT "NO"
END IF
END
```

Python

```
n = int(input())
k = 0
while n % 4 == 0:
    k = k + n // 4
    n = n // 4
if n > 0:
    print(k)
else:
    print("NO")
```

Алгоритмический язык

```
алг
нач
  цел n, k
  ввод n
  k := 0
  нц пока mod(n, 4)=0
    k := k + div(n, 4)
    n := div(n, 4)
  кц
  если n > 0
    то вывод k
    иначе вывод "NO"
  все
кон
```

Паскаль

```
var n, k: integer;
begin
  read(n);
  k := 0;
  while n mod 4 = 0 do begin
    k := k + n div 4;
    n := n div 4;
  end;
  if n > 0 then
    writeln(k)
  else
    writeln('NO')
end.
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int n, k;
    cin >> n;
    k = 0;
    while (n % 4 == 0) {
        k = k + n / 4;
        n = n / 4;
    }
    if (n > 0)
        cout << k << endl;
    else
        cout << "NO" << endl;
    return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 64.
2. Приведите пример числа, при вводе которого приведённая программа, несмотря на ошибки, выведет правильный ответ.
3. Найдите допущенные программистом ошибки и исправьте их. Исправление ошибки должно затрагивать только строку, в которой находится ошибка. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

Известно, что в тексте программы можно исправить ровно две строки так, чтобы она стала работать правильно.

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

Обратите внимание на то, что требуется найти ошибки в имеющейся программе, а не написать свою, возможно, использующую другой алгоритм решения.

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

Решение использует запись программы на Паскале. Допускается использование программы на других языках.

1. При вводе числа 64 программа выведет число 21.

2. Примеры чисел, при вводе которых программа выводит верный ответ: 1, 4. Других чисел нет.

Комментарий для экспертов. Цикл заканчивается, когда значение переменной *n* перестаёт делиться на 4. Так как в переменной *K* вместо количества итераций цикла подсчитывается сумма делимых значений *N*, то в итоге *K* содержит правильный ответ только при $N = 4^0$ и $N = 4^1$.

Таким образом, программа выводит корректное и существующее значение *K*, только если введено число 1 или 4. Экзаменуемому достаточно указать любое из этих чисел.

3. В программе есть две ошибки.

Первая ошибка: неверное изменение счётчика.

Строка с ошибкой:

```
k := k + n div 4;
```

Верное исправление:

```
k := k + 1;
```

Вторая ошибка: неверное условие при печати результата.

Строка с ошибкой:

```
if n > 0 then
```

Верное исправление:

```
if n = 1 then
```

Пояснение для эксперта

После исправления первой ошибки в результате выполнения цикла значение переменной *n* будет равно $n_0/(4^k)$, где n_0 – введённое пользователем значение, *k* – максимальный показатель степени, при котором 4^k является делителем числа n_0. Число n_0 является степенью числа 4, если $n_0 = 4^k$, т.е. $n_0/(4^k) = 1$.

В программах на других языках ошибочные строки и их исправления аналогичны.

Незначительной опиской, не влияющей на оценку, следует считать отсутствие служебных слов и знаков после содержательной части исправления

Указания по оцениванию	Баллы
В задаче требуется выполнить **три** действия. 1. Указать результат программы при данном вводе. Это действие считается выполненным, если указан верный результат работы программы при заданных входных данных. Экзаменуемый не обязан объяснять, как получен этот результат, достаточно указать верное число. 2. Указать пример ввода, при котором программа выводит верный ответ. Это действие считается выполненным, если указан пример числа, при вводе которого выводится верный показатель степени (1 или 4). Ученик не обязан указывать, что будет выведено, и объяснять, как работает программа. 3. Найти и исправить ошибки в программе. Это действие считается выполненным, если верно указаны обе ошибки и предложены верные варианты исправления, при этом никакие верные строки программы не указаны в качестве неверных. В исправленной строке допускаются незначительные синтаксические ошибки (лишние или пропущенные знаки препинания, неточные написания служебных слов языка). 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 три 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. 1. Выполнены два первых действия, найдена и исправлена одна ошибка в программе, ни одна верная строка не названа ошибочной. 2. Выполнены два первых действия, найдены и исправлены две ошибки в программе, одна верная строка названа ошибочной. 3. Выполнено одно из первых двух действий, найдены и исправлены две ошибки в программе, ни одна верная строка не названа ошибочной	2
Не выполнены условия, позволяющие поставить 2 или 3 балла. При этом имеет место один из следующих случаев. 1. Выполнены два первых действия. При этом несущественно, насколько правильно выполнено третье действие.	1

2. Найдены и исправлены две ошибки в программе, не более чем одна верная строка названа ошибочной. При этом несущественно, насколько правильно выполнены действия 1 и 2. 3. Выполнено одно из двух первых действий. Исправлена одна из двух ошибок. Не более чем одна верная строка названа ошибочной	
Не выполнены условия, позволяющие поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

25 Дан целочисленный массив из 30 элементов. Элементы массива могут принимать натуральные значения от 1 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит минимум среди элементов массива, кратных 4, а затем заменяет каждый элемент, кратный 4, на число, равное найденному минимуму. Гарантируется, что хотя бы один такой элемент в массиве есть. В качестве результата необходимо вывести изменённый массив, каждый элемент выводится с новой строчки.

Например, для исходного массива из шести элементов:

12
5
8
5
8
16

программа должна вывести следующий массив

8
5
8
5
8
8

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

Бейсик

```
CONST N AS INTEGER = 30
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 = 30
for i in range(0, n):
    a.append(int(input()))
...
```

Алгоритмический язык

```
алг
нач
    цел N = 30
    целтаб a[1:N]
    цел i, j, k
    нц для i от 1 до N
        ввод a[i]
    кц
    ...

кон
```

Паскаль

```
const
    N = 30;
var
    a: array [1..N] of longint;
    i, j, k: longint;
begin
    for i := 1 to N do
        readln(a[i]);
    ...

end.
```

C++

```
#include <iostream>
using namespace std;
const int N = 30;
int main() {
    long a[N];
    long i, j, k;
    for (i = 0; i < N; i++)
        cin >>  a[i];
    ...
    return 0;
}
```

В качестве ответа Вам необходимо привести фрагмент программы, 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). 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 := 10000;
for i := 1 to N do
  if (a[i] mod 4 = 0) and (a[i] < k) then
    k := a[i];
for i := 1 to N do begin
  if (a[i] mod 4 = 0) then
    a[i] := k;
  writeln(a[i]);
end;
```

На Алгоритмическом языке

```
k := 10000
нц для i от 1 до N
  если mod(a[i], 4) = 0 и a[i] < k
  то
    k := a[i]
  все
кц
нц для i от 1 до N
  если mod(a[i], 4) = 0
  то
    a[i] := k
  все
  вывод a[i], нс
кц
```

На языке Бейсик

```
K = 10000
FOR I = 1 TO N
  IF A(I) MOD 4 = 0 AND A(I) < K THEN
    K = A(I)
  END IF
NEXT I
FOR I = 1 TO N
  IF A(I) MOD 4 = 0 THEN
    A(I) = K
  END IF
  PRINT A(I)
NEXT I
```

На языке С++

```
k = 10000;
for (i = 0; i < N; i++)
  if (a[i] % 4 == 0 && a[i] < k)
    k = a[i];
for (i = 0; i < N; i++) {
  if (a[i] % 4 == 0)
    a[i] = k;
  cout << a[i] << endl;
}
```

На языке Python

```
k = 10000
for i in range(0, n):
    if (a[i] % 4 == 0 and a[i] < k):
        k = a[i]
for i in range(0, n):
    if (a[i] % 4 == 0):
        a[i] = k
    print(a[i])
```

Указания по оцениванию	**Баллы**
Общие указания. 1. В алгоритме, записанном на языке программирования, допускается наличие отдельных синтаксических ошибок, не искажающих замысла автора программы. 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. 4. Допускается формат вывода массива, отличный от указанного, например в строчку	
Предложен правильный алгоритм, который изменяет исходный массив и выводит в качестве результата изменённый массив	2

Не выполнены условия, позволяющие поставить 2 балла. При этом предложено в целом верное решение, содержащее не более одной ошибки из числа следующих: 1) в цикле происходит выход за границу массива; 2) не инициализируется или неверно инициализируется минимум; 3) неверно осуществляется проверка делимости на 4; 4) проверяется делимость на 4 не элемента массива, а его индекса; 5) в сравнении с минимумом перепутаны знаки «больше» и «меньше»; 6) сравнение с минимумом производится для индекса элемента массива, а не для его значения; 7) неверно составлено логическое условие (например, используется or вместо and); 8) исходный массив не изменяется; 9) изменяются не все требуемые элементы (например, только первый или последний из них); 10) отсутствует вывод ответа, или ответ выводится не полностью (например, только один элемент массива ввиду пропущенного цикла вывода элементов или операторных скобок); 11) используется переменная, не объявленная в разделе описания переменных; 12) не указано или неверно указано условие завершения цикла; 13) индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно	1
Ошибок, перечисленных в п. 1–13, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии в явном или неявном виде цикла поиска нужного элемента)	0
Максимальный балл	*2*

26 Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежат две кучи камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в одну из куч (по своему выбору) **один** камень или увеличить количество камней в куче в **два раза**. Например, пусть в одной куче 10 камней, а в другой 7 камней; такую позицию в игре будем обозначать (10, 7). Тогда за один ход можно получить любую из четырёх позиций: (11, 7), (20, 7), (10, 8), (10, 14). Для того чтобы делать ходы, у каждого игрока есть неограниченное количество камней.

Игра завершается в тот момент, когда суммарное количество камней в кучах становится не менее 75. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, при которой в кучах будет 75 или больше камней.

В начальный момент в первой куче было пять камней, во второй куче – S камней; $1 \le S \le 69$.

Будем говорить, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника. 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.

Выполните следующие задания

Задание 1

а) Укажите все такие значения числа S, при которых Петя может выиграть за один ход.

б) Известно, что Ваня выиграл своим первым ходом после неудачного первого хода Пети. Укажите минимальное значение S, когда такая ситуация возможна.

Задание 2

Укажите такое значение S, при котором у Пети есть выигрышная стратегия, причём одновременно выполняются два условия:

– Петя не может выиграть за один ход;
– Петя может выиграть своим вторым ходом независимо от того, как будет ходить Ваня.

Для указанного значения S опишите выигрышную стратегию Пети.

Задание 3

Укажите значение S, при котором одновременно выполняются два условия:

– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети;
– у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Для указанного значения S опишите выигрышную стратегию Вани.

Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы).
В узлах дерева указывайте позиции, на рёбрах рекомендуется указывать ходы. Дерево не должно содержать партии, невозможные 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)

Задание 1

а) Петя может выиграть при $35 \le S \le 69$.

б) $S = 18$.

Задание 2

Возможное значение *S*: 34. В этом случае Петя, очевидно, не может выиграть первым ходом. Однако он может получить позицию (6, 34). После хода Вани может возникнуть одна из четырёх позиций: (7, 34), (12, 34), (6, 35), (6, 68). В каждой из этих позиций Петя может выиграть одним ходом, удвоив количество камней во второй куче.
Замечание для проверяющего. Ещё одно возможное значение *S* для этого задания – число 32. В этом случае Петя первым ходом должен удвоить количество камней в меньшей куче и получить позицию (5 * 2, 32) = (10, 32). При такой позиции Ваня не может выиграть первым ходом, а после любого хода Вани Петя может выиграть, удвоив количество камней в большей куче. Достаточно указать одно значение *S* и описать для него выигрышную стратегию.

Задание 3

Возможное значение *S*: 33. После первого хода Пети возможны позиции: (6, 33), (10, 33), (5, 34), (5, 66). В позициях (10, 33) и (5, 66) Ваня может выиграть первым ходом, удвоив количество камней во второй куче. Из позиций (6, 33) и (5, 34) Ваня может получить позицию (6, 34). Эта позиция разобрана в п. 2. Игрок, который её получил (теперь это Ваня), выигрывает своим вторым ходом.
Замечание для проверяющего. Ещё одно возможное значение *S* для этого задания – число 31. После первого хода Пети возможны позиции: (6, 31), (10, 31), (5, 32), (5, 62). Из позиции (6, 31) Ваня может получить позицию (12, 31) и независимо от ответного хода Пети выиграть своим следующим ходом, удвоив количество камней во второй куче. Из позиций (10, 31) и (5, 32) Ваня может получить позицию (10, 32). Эта позиция разобрана в замечании к заданию 2. В позиции (5, 62) Ваня может выиграть первым ходом, удвоив количество камней во второй куче.

В таблице изображено дерево возможных партий (и только их) при описанной стратегии Вани для S = 33. Заключительные позиции (в них выигрывает Ваня) выделены жирным шрифтом. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

	Положения после очередных ходов			
Исходное положение	1-й ход Пети (разобраны все ходы, указана полученная позиция)	1-й ход Вани (только ход по стратегии, указана полученная позиция)	2-й ход Пети (разобраны все ходы, указана полученная позиция)	2-й ход Вани (только ход по стратегии, указана полученная позиция)
(5, 33) Всего: 38	(5, 33+1) = (5, 34) Всего: 39	(5+1, 34) = (6, 34) Всего: 40	(6+1, 34) = (7, 34) Всего: 41	(7, 34*2) = (7, 68) **Всего: 75**
			(6, 34+1) = (6, 35) Всего: 41	(6, 35*2) = (6, 70) **Всего: 76**
			(6*2, 34) = (12, 34) Всего: 46	(12, 34*2) = (12, 68) **Всего: 80**
			(6, 34*2) = (6, 68) Всего: 74	(6, 68*2) = (6, 136) **Всего: 142**
	(5+1, 33) = (6, 33) Всего: 39	(6, 33+1) = (6, 34) Всего: 40	(6+1, 34) = (7, 34) Всего: 41	(7, 34*2) = (7, 68) **Всего: 75**
			(6, 34+1) = (6, 35) Всего: 41	(6, 35*2) = (6, 70) **Всего: 76**
			(6*2, 34) = (12, 34) Всего: 46	(12, 34*2) = (12, 68) **Всего: 80**
			(6, 34*2) = (6, 68) Всего: 74	(6, 68*2) = (6, 136) **Всего: 142**
	(5*2, 33) = (10, 33) Всего: 43	(10, 33*2) = (10, 66) **Всего: 76**		
	(5, 33*2) = (5, 66) Всего: 71	(5, 66*2) = (5, 132) **Всего: 137**		

Примечание для эксперта. Дерево всех партий может быть также изображено в виде ориентированного графа – так, как показано на рисунке, или другим способом. Важно, чтобы множество полных путей в графе находилось во взаимно однозначном соответствии со множеством партий, возможных при описанной в решении стратегии.

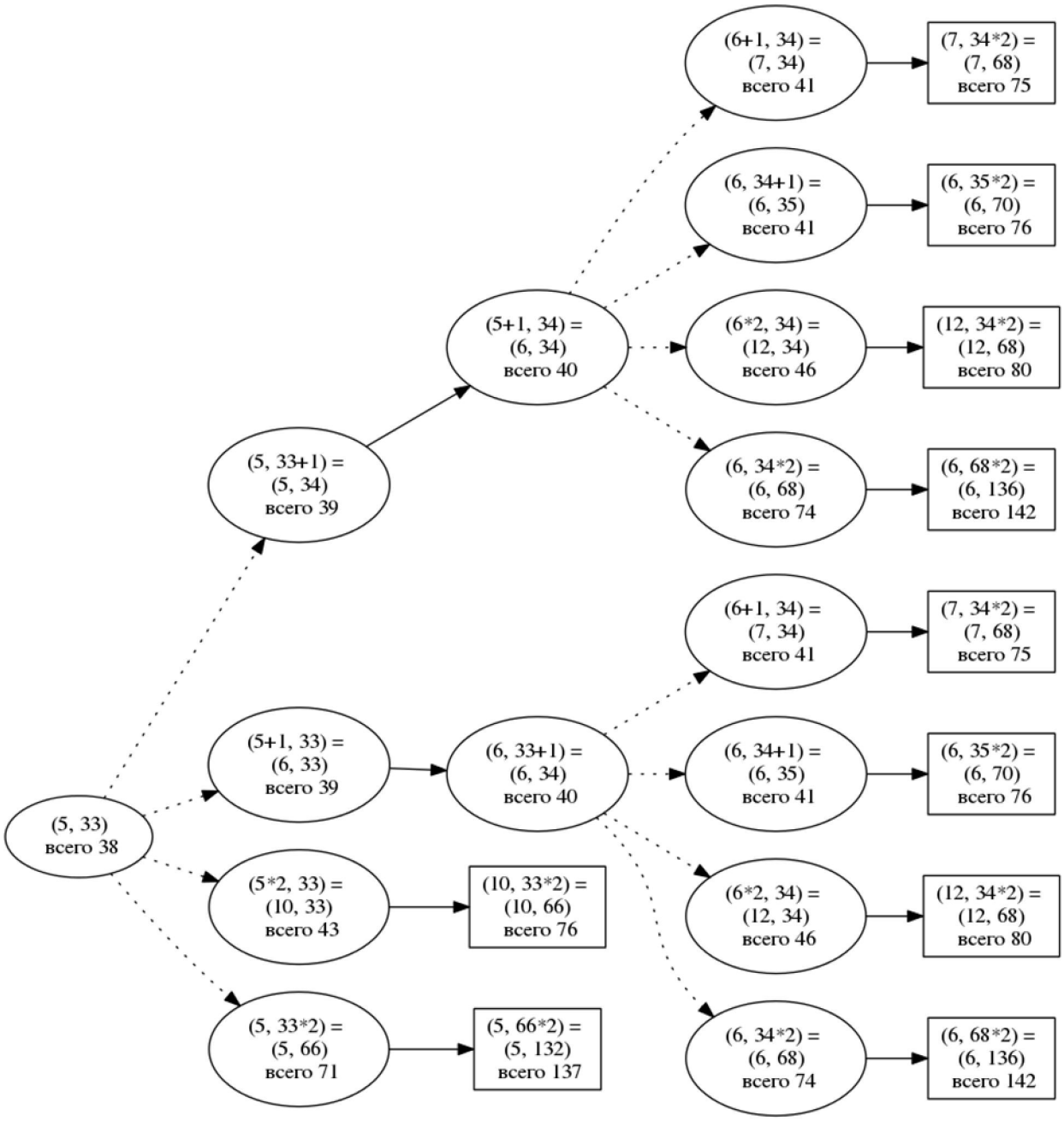

Рис. 1. Дерево всех партий, возможных при Ваниной стратегии. Ходы Пети показаны пунктиром; ходы Вани – сплошными линиями. Прямоугольником обозначены позиции, в которых партия заканчивается.

Замечание для проверяющего. Не является ошибкой указание только одного заключительного хода выигрывающего игрока в ситуации, когда у него есть более одного выигрышного хода

Указания по оцениванию	**Баллы**
В задаче требуется выполнить три задания. Их трудность возрастает. Количество баллов в целом соответствует количеству выполненных заданий (подробнее см. ниже). Ошибка в решении, не искажающая основного замысла и не приведшая к неверному ответу, например арифметическая ошибка при вычислении количества камней в заключительной позиции, при оценке решения не учитывается. Задание 1 выполнено, если выполнены оба пункта: а) и б), т.е. для п. а) перечислены все значения S, удовлетворяющие условию (и только они), для п. б) указано верное значение S (и только оно). Задание 2 выполнено, если правильно указана позиция, выигрышная для Пети, и описана соответствующая стратегия Пети – так, как это сделано в примере решения, или другим способом, например с помощью дерева всех возможных при выбранной стратегии Пети партий (и только их). Задание 3 выполнено, если правильно указана позиция, выигрышная для Вани, и построено дерево всех возможных при Ваниной стратегии партий (и только их). Во всех случаях стратегии могут быть описаны так, как это сделано в примере решения, или другим способом	
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 3, 2 или 1 балл	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, находящихся на расстоянии не меньше чем 3 (разница в индексах элементов пары должна быть 3 или более, порядок элементов в паре неважен). Необходимо определить количество таких пар, для которых произведение элементов делится на 23.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел N ($3 \le N \le 1000$). В каждой из последующих N строк записано одно целое положительное число, не превышающее 10 000.

В качестве результата программа должна вывести одно число: количество пар элементов, находящихся в последовательности на расстоянии не меньше чем 3, в которых произведение элементов кратно 23.

Пример входных данных:

```
6
46
2
3
5
4
23
```

Пример выходных данных для приведённого выше примера входных данных:

```
5
```

Пояснение. Из шести заданных элементов с учётом допустимых расстояний между ними можно составить 6 произведений: 46·5, 46·4, 46·23, 2·4, 2·23, 3·23. Из них на 23 делятся 5 произведений.

Требуется написать эффективную по времени и 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 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 23, если хотя бы один из сомножителей делится на 23. При вводе чисел можно подсчитывать количество чисел, кратных 23, не считая 3 последних. Обозначим их *n*23. *Примечание для проверяющего*. Сами числа, кроме 3 последних, при этом можно не хранить. Очередное считанное число будем рассматривать как возможный правый элемент искомой пары. Если очередное считанное число делится на 23, то к ответу следует прибавить количество чисел до него, не считая 3 последних (включая считанное). Если очередное считанное число на 23 не делится, то к ответу следует прибавить *n*23. Чтобы построить программу, эффективную по памяти, заметим, что, поскольку при обработке очередного элемента входных данных используются значения, находящиеся на 3 элемента ранее, достаточно хранить только 3 последних элемента или информацию о них. Ниже приведена реализующая описанный алгоритм программа на языке Паскаль (использована версия PascalABC)

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

```
const s = 3; {требуемое расстояние между элементами}
var
  n: longint;
  a: array[1..s] of longint; {хранение последних s значений}
  a_: longint; {очередное значение}
  n23: longint; {количество делящихся на 23 элементов,
           не считая s последних}
  cnt: longint; {количество искомых пар}
  i, j: longint;
begin
  readln(n);
  {Ввод первых s чисел}
  for i:=1 to s do
    readln(a[i]);
  {Ввод остальных значений, подсчет искомых пар}
  cnt := 0;
  n23 := 0;
  for i := s + 1 to n do
  begin
    if a[1] mod 23 = 0 then
      n23 := n23 + 1;
    readln(a_);
    if a_ mod 23 = 0 then
      cnt := cnt + i - s
    else
      cnt := cnt + n23;
    {сдвигаем элементы вспомогательного массива влево}
    for j := 1 to s - 1 do
      a[j] := a[j + 1];
    a[s] := a_ {записываем текущий элемент в конец массива}
  end;
  writeln(cnt)
end.
```

Комментарии для проверяющего

1. При таком решении хранятся только последние 3 прочитанных элемента. Таким образом, используемая память не зависит от длины последовательности. Время обработки очередного числа фиксировано, т.е. 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 в 4 балла.

В таких версиях Паскаля, как PascalABC или Delphi, тип longint может быть заменён на тип integer. В большинстве версий языков C\C++ также можно использовать тип int.

Программа может быть и ещё более эффективной, если на каждом шаге не сдвигать элементы вспомогательного массива, а записывать i-й считанный

элемент в элемент с индексом `i mod 3` (Паскаль) или `i % 3` (Python), ведя нумерацию обоих индексов с нуля. Учёту подлежит элемент с этим же индексом (именно он находится на расстоянии *s* от *i*-го и будет заменён на него). Кроме того, при нумерации индексов элементов с нуля меняется одна из формул для подсчёта.

Такая программа на языке Python приведена ниже (пример 2).

Все подобные программы оцениваются, исходя из максимального балла – 4 (см. критерии).

Вместо последних 3 элементов можно хранить и 3 счётчика: количество делящихся на 23 среди всех считанных чисел, всех считанных чисел без последнего, всех считанных чисел без 2 последних – и также сдвигать их после очередного шага. Такая программа приведена на языке C++ (пример 3). В этом же примере вместо вспомогательного массива длиной 3 используются 3 переменные.

2. Возможно решение, основанное на описанных идеях, однако предварительно сохраняющее элементы последовательности в массив. Такое решение эффективно по времени, но неэффективно по памяти. Оно оценивается, исходя из максимального балла – 3 (см. критерии).

3. Решение, неэффективное ни по времени, ни по памяти, запоминает входную последовательность в массиве, после чего явно перебирает все возможные пары. Такое решение оценивается, исходя из максимального балла – 2 (см. критерии).

Пример 2. Программа на языке Python. Программа эффективна по времени и памяти

```
s = 3
a = [0]*s
n = int(input())
for i in range(s):
  a[i] = int(input())
cnt = 0
n23 = 0
for i in range(s, n):
    k = i % s
    if a[k] % 23 == 0:
        n23 = n23 + 1
    a_ = int(input())
    if   a_ % 23 == 0:
        cnt = cnt + i - s + 1
    else:
        cnt = cnt + n23
    a[i % s] = a_
print(cnt)
```

Пример 3. Программа на языке С++. Программа эффективна по времени и памяти

```
#include <iostream>
using namespace std;
int main()
{
  int s = 3; //требуемое расстояние между элементами
  int n;
  int n1 = 0, n2 = 0, n3 = 0; //хранение последних s счетчиков
  int a_;  // очередное значение
  int cnt; // количество искомых пар
  cin >> n;
  cnt = 0;
  for (int i = 0; i < n; ++i)
  {
    cin >> a_; // считано очередное значение
    if (i >= s)
    {
        if (a_ % 23 == 0)
           cnt += i - s + 1;
        else
           cnt += n3;
    }
    //сдвигаем элементы счетчиков
    n3 = n2;
    n2 = n1;
    //обновляем счетчик кратных 23
    if (a_ % 23 == 0)
        n1 += 1;
  }
  cout << cnt;
  return 0;
}
```

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 без программы оценивается в 0 баллов.	
Программа правильно работает для любых входных данных произвольного размера при условии исправления в ней не более трёх синтаксических ошибок из приведённого ниже списка допустимых ошибок. 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, пропущено или написано лишнее 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. Программа работает правильно для любых входных данных произвольного размера при условии исправления в ней не более пяти синтаксических ошибок из приведённого в критериях на 4 балла списка и не более одной ошибки из приведённого ниже списка содержательных ошибок. Время работы пропорционально количеству введённых чисел. Допускается наличие не более **одной** содержательной (не являющейся синтаксической) ошибки 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) неверная инициализация или её отсутствие там, где она необходима; 3) используется неверный тип данных; 4) использована одна переменная (или константа) вместо другой; 5) используется один знак операции вместо другого; 6) используется одно зарезервированное слово языка программирования вместо другого; 7) неверно используется условный оператор, например else относится не к тому условию; 8) отсутствует вывод ответа, или выводится значение не той переменной;	3

9) выход за границу массива; 10) неверно расставлены операторные скобки. 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)	
Пример 4. Программа на языке Паскаль. Программа эффективна по времени и неэффективна по памяти	

`const s = 3; {требуемое расстояние между элементами}`
`var`
`  n: longint;`
`  a: array[1..1000] of longint;`
`  n23: longint;`
`{количество делящихся на 23 элементов,  не считая s последних}`
`  cnt: longint; {количество искомых пар}`
`  i, j: longint;`
`begin`
`  readln(n);`
`  {Ввод первых s чисел}`
`  for i:=1 to s do`
`    readln(a[i]);`
`  {Ввод остальных значений, подсчет искомых пар}`
`  cnt := 0;`
`  n23 := 0;`
`  for i := s + 1 to n do`
`  begin`
`    readln(a[i]);`
`    if a[i - s] mod 23 = 0 then`
`      n23 := n23 + 1;`
`    if a[i] mod 23 = 0 then`
`      cnt := cnt + i - s`
`    else`
`      cnt := cnt + n23;`
`  end;`
`  writeln(cnt)`
`end.`	

<table>
<tr><td>Не выполнены условия, позволяющие поставить 3 или 4 балла.
Программа работает верно, эффективно по времени при условии исправления не более трёх содержательных ошибок, описанных в критериях на 3 балла, и не более девяти синтаксических ошибок, указанных в критериях на 4 балла.
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 с учётом допустимого расстояния между ними. Пример фрагмента соответствующей программы на языке Паскаль:
<pre>
cnt := 0;
for i := 1 to N - s do
  for j := i + s to N do
    if a[i] * a[j] mod 23 = 0 then
      cnt := cnt + 1;
writeln(cnt)
</pre>
Не допускается выставление 2 баллов за реализацию переборного алгоритма, содержащего любую логическую ошибку, например ошибку, приводящую к выходу индексов за границы массива, или ошибку, когда учитываются произведения вида <code>a[i]*a[i]</code>, или пары считаются дважды, или неверно учитывается расстояние между индексами элементов пары</td><td>2</td></tr>
<tr><td>Не выполнены условия, позволяющие поставить 2, 3 или 4 балла.
При этом в программе должны присутствовать два обязательных элемента, возможно, реализованных с ошибками:
1) проверка делимости (в явной или неявной форме) элементов входной последовательности на заданное число;
2) проверка или учёт того, что расстояние между элементами искомой пары должно быть не меньше заданного</td><td>1</td></tr>
<tr><td>Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла</td><td>0</td></tr>
<tr><td>Максимальный балл</td><td>4</td></tr>
</table>