Ответы к заданиям

№ задания	Ответ
1	8
2	xwzy
3	45
4	4
5	111
6	33
7	15
8	95
9	4
10	405
11	7412131
12	224
13	14
14	110
15	9
16	38
17	12
18	61
19	202
20	393
21	31
22	14
23	36

Критерии оценивания заданий с развёрнутым ответом

24 Требовалось написать программу, которая получает на вход натуральное число N, не превосходящее 10^9, и выводит число, равное количеству цифр 2 в десятичной записи числа N. Программист написал программу неправильно. Ниже эта написанная им программа для Вашего удобства приведена на пяти языках программирования.

Бейсик

```
DIM N AS LONG
DIM R, d AS INTEGER
INPUT N
R = 0
WHILE N > 0
  d = N MOD 10
  IF d <> 2 THEN
    R = R + 1
  END IF
  N = N \ 10
WEND
PRINT d
END
```

Python

```
N = int(input())
R = 0
while N > 0:
    d = N % 10
    if d != 2:
       R = R + 1
    N = N // 10
print(d)
```

Алгоритмический язык

```
алг
нач
  цел N, R, d
  ввод N
  R := 0
  нц пока N > 0
    d := mod(N, 10)
    если d <> 2 то
      R := R + 1
    все
    N := div(N, 10)
  кц
  вывод d
кон
```

Паскаль

```
var N: longint;
  R, d: integer;
begin
  readln(N);
  R := 0;
  while N > 0 do begin
    d := N mod 10;
    if d <> 2 then
      R := R + 1;
    N := N div 10;
  end;
  writeln(d);
end.
```


C++

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

int main()
{
  long int N;
  int R, d;
  cin >> N;
  R = 0;
  while (N > 0) {
    d = N % 10;
    if (d != 2) {
      R = R + 1;
    }
    N = N / 10;
  }
  cout << d << endl;
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 324.
2. Приведите пример входного числа N, при котором приведённая программа, несмотря на ошибки, выдаёт верный 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 нужно исправить не более двух строк 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 любом из четырёх других языков программирования.
1. Программа выведет число 3.
Примечание для эксперта. Программа выводит значение первой слева цифры числа *N*.
2. Программа выдаёт правильный ответ, например, при *N* = 3222.
3. В программе есть две ошибки.
Первая ошибка: неверная проверка условия увеличения счетчика – переменной *R*.
Строка с ошибкой:
`if d <> 2 then`
Верное исправление:
`if d = 2 then`
Вторая ошибка: вместо значения переменной *R* выводится значение переменной d.
Строка с ошибкой:
`writeln(d);`
Верное исправление:
`writeln(R);`	

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **четыре** действия: 1) указать, что выведет программа при конкретном входном числе; 2) указать пример входного числа, при котором программа выдаёт верный 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 из четырёх необходимых действий	1
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 целые значения от 0 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит сумму элементов массива, больших 100 и при этом не кратных 4, а затем заменяет каждый такой элемент на число, равное найденной сумме. 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:
101
128
6
105
4
18
программа должна вывести следующий массив:
206
128
6
206
4
18
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 = 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 := 0;
for i := 1 to N do
  if (a[i] > 100) and (a[i] mod 4 <> 0) then
    k := k + a[i];
for i := 1 to N do begin
  if (a[i] > 100) and (a[i] mod 4 <> 0) then
    a[i] := k;
  writeln(a[i]);
end;
```

На Алгоритмическом языке

```
k := 0
нц для i от 1 до N
  если a[i] > 100 и mod(a[i], 4) <> 0
  то
    k := k + a[i]
  все
кц
нц для i от 1 до N
  если a[i] > 100 и mod(a[i], 4) <> 0
  то
    a[i] := k
  все
  вывод a[i], нс
кц
```

На языке Бейсик

```
K = 0
FOR I = 1 TO N
  IF A(I) > 100 AND A(I) MOD 4 <> 0 THEN
    K = K + A(I)
  END IF
NEXT I
FOR I = 1 TO N
  IF A(I) > 100 AND A(I) MOD 4 <> 0 THEN
    A(I) = K
  END IF
  PRINT A(I)
NEXT I
```

На языке C++

```
k = 0;
for (i = 0; i < N; i++)
  if (a[i] > 100 && a[i] % 4 != 0)
    k = k + a[i];
for (i = 0; i < N; i++) {
  if (a[i] > 100 && a[i] % 4 != 0)
    a[i] = k;
  cout << a[i] << endl;
}
```

На языке Python

```
k = 0
for i in range(0, n):
    if (a[i] > 100 and a[i] % 4 != 0):
        k = k + a[i]
for i in range(0, n):
    if (a[i] > 100 and 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 (в том числе без разделителей)	

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 сумма найденных элементов; 3) неверно осуществляется проверка делимости на 4; 4) проверяется делимость на 4 не элемента массива, а его индекса; 5) неверно выбрана операция отношения для сравнения с границей диапазона; 6) сравнение с границей диапазона производится для индекса элемента массива, а не для его значения; 7) неверно составлено логическое условие (например, используется or вместо and); 8) не вычисляется или неверно накапливается сумма найденных элементов; 9) исходный массив не изменяется; 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 подсчёта суммы нужных элементов)	0
Максимальный балл	*2*

26 Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один** или **четыре** камня либо увеличить количество камней в куче **в пять раз**. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 19 или 75 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 68.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший кучу, в которой будет 68 или больше камней.
В начальный момент в куче было S камней; $1 \le S \le 67$.

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. Во всех случаях обосновывайте свой ответ.

Задание 1

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

б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

Задание 2

Укажите два таких значения S, при которых 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 каждого 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). На рёбрах дерева указывайте, кто делает ход; в узлах – количество камней в куче.

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, если $S = 14, \ldots 67$.

б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче $S = 13$ камней. Тогда после первого хода Пети в куче будет 14, 17 или 65 камней. Во всех случаях Ваня увеличивает количество камней в 5 раз и выигрывает за один ход.

Замечание для проверяющего. В задаче не требуется указать **все** выигрышные стратегии. Если в работе выпускника, как в приведённом примере, просто указано, что Ваня всегда увеличивает в 5 раз количество камней, – это не ошибка.

Задание 2

Возможные значения S: 9, 12. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 13 камней. Эта позиция разобрана в п. 1б. В ней игрок, который будет ходить (теперь это Ваня), выиграть не может, а его противник (т.е. Петя) следующим ходом выиграет.

Задание 3

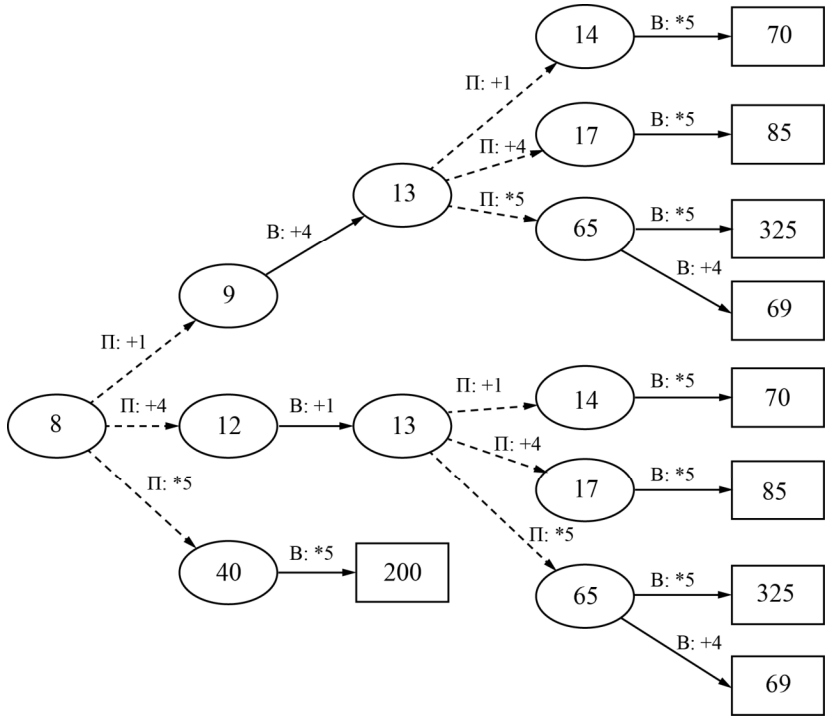
Возможные значения S: 8, 11.

Например, для $S = 8$ после первого хода Пети в куче будет 9, 12 или 40 камней. Если в куче станет 40 камней, Ваня увеличит количество камней в 5 раз и выиграет первым ходом. Ситуация, когда в куче 9 или 12 камней, разобрана в п. 2. В этой ситуации игрок, который будет ходить (теперь это Ваня), выигрывает своим вторым ходом.

В таблице изображено дерево возможных партий (и только их) при описанной стратегии Вани для значения $S = 8$. При выбранной стратегии на последнем ходе Ваня увеличивает в 5 раз количество камней, хотя возможны и другие выигрышные стратегии. Для второго возможного значения дерево строится аналогично. Заключительные позиции (в них выигрывает Ваня) подчёркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

Примечание для проверяющего. Здесь для полноты картины указаны два возможных значения S. По условию задачи достаточно указать одно из них.

Исходное положение	Положения после очередных ходов			
	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)
8	8 +1 = **9**	9 + 4 = **13**	13 + 1 = **14**	14 * 5 = **70**
			13 + 4 = **17**	17 * 5 = **85**
			13 * 5 = **65**	65 * 5 = **325**
				65 + 4 = **69**
	8 + 4 = **12**	12 + 1 = **13**	13 + 1 = **14**	14 * 5 = **70**
			13 + 4 = **17**	17 * 5 = **85**
			13 * 5 = **65**	65 * 5 = **325**
				65 + 4 = **69**
	8 * 5 = **40**	40 * 5 = **200**		



Дерево всех партий, возможных при Ваниной стратегии.
Прямоугольником обозначены позиции, в которых партия заканчивается

Замечание для проверяющего. На рисунке для наглядности ходы Пети показаны пунктиром, а заключительные позиции выделены прямоугольником. Это не является обязательным для экзаменуемых. Также не является ошибкой указание только одного заключительного хода Вани в ситуации, когда у него есть два заключительных выигрышных хода

Указания по оцениванию	**Баллы**
В задаче от выпускника 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). Если хотя бы один из этих пунктов не выполнен или выполнен с ошибкой (кроме оговоренных выше ошибок), задание считается невыполненным. Задание 2 выполнено, если правильно указаны обе позиции, выигрышные для Пети, и описана соответствующая стратегия Пети – так, как это написано 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 2 или 3 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 Дана последовательность N целых положительных чисел. Рассматриваются все пары элементов последовательности, разность которых чётна, и в этих парах, по крайней мере, одно из чисел пары делится на 19. Порядок элементов в паре неважен. Среди всех таких пар нужно найти и вывести пару с максимальной суммой элементов. Если одинаковую максимальную сумму имеет несколько пар, можно вывести любую из них. Если подходящих пар в последовательности нет, нужно вывести два нуля.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел N ($2 \le N \le 10\,000$). В каждой из последующих N строк записано одно натуральное число, не превышающее 10 000.

Пример входных данных:

```
5
38
12
57
16
57
```

Пример выходных данных для приведённого выше примера входных данных:

```
57 57
```

Пояснение. Из данных пяти чисел можно составить три различные пары, удовлетворяющие условию: (38, 12), (38, 16), (57, 57). Наибольшая сумма получается в паре (57, 57). Эта пара допустима, так как число 57 встречается в исходной последовательности дважды.

Напишите эффективную по времени и памяти программу для решения этой 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 Кбайт 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 или только по памяти, – 3 балла.

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

Вы можете сдать **одну** или **две** программы решения задачи. Если Вы сдадите две программы, каждая из них будет оцениваться независимо от другой, итоговой станет **бóльшая** из двух оценок.

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

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Разность двух чисел чётна, если эти числа имеют одинаковую чётность. Будем вводить элементы последовательности по одному и хранить четыре элемента: максимальные чётный m_0 и нечётный m_1 элементы всей последовательности и максимальные чётный и нечётный элементы, кратные $p = 19$ (m_{p0} и m_{p1} соответственно). При этом у чисел, кратных p, будет «приоритет»: если вновь найденный общий максимум последовательности (с учётом чётности) кратен p, это число будет записано в качестве максимума среди кратных p (m_{p0} или m_{p1}). В общий максимум (m_0 или m_1) оно сможет перейти, только если среди кратных p (m_{p0} или m_{p1}) появится большее или такое же значение. После ввода всех элементов нужно найти значения $m_{p0} + m_0$, $m_{p1} + m_1$ и выбрать пару с большей суммой. При этом необходимо убедиться, что эти максимумы действительно выбраны из последовательности, а не записаны при инициализации нулевыми или отрицательными значениями. Ниже приведены программы на языках Pascal (использована версия PascalABC) и Алгоритмическом, реализующие этот алгоритм

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

```
const p = 19;
var
    N: integer;     {количество чисел}
    a: integer;     {очередное число}
    d: integer;     {чётность очередного числа}

    m: array [0..1] of integer;  {чётный и нечётный максимумы}
    mp: array [0..1] of integer;
                     {чётный и нечётный максимумы, кратные p}
    x, y: integer;     {ответ – пара чисел}
    i: integer;


begin
   m[0] := 0; m[1] := 0;
   mp[0] := 0; mp[1] := 0;
   x := 0; y := 0;
   readln(N);
   for i := 1 to N do
     begin
       readln(a);
         d := a mod 2;
         if (a mod p = 0)and(a >= mp[d]) then
                              {нестрогое сравнение!}
           begin
             if mp[d] > m[d] then {допустимо нестрогое сравнение}
                 m[d] := mp[d];
             mp[d] := a
          end
          else if a > m[d] then {допустимо нестрогое сравнение}
             m[d] := a
     end;
   if (mp[0] > 0) and (m[0] > 0) then
       x := mp[0]; y := m[0];
   if (mp[1] > 0) and (m[1] > 0) and (mp[1] + m[1] > x + y) then
       x := mp[1]; y := m[1];
   writeln(x, ' ', y)
end.
```

Пример 1а. Программа на Алгоритмическом языке. Программа эффективна по времени и памяти

```
алг задача
нач
  цел p = 19| делитель из условия задачи
  цел N | количество чисел
  цел a | очередное число
  цел d | чётность очередного числа
  целтаб m[0:1] |чётный и нечётный максимумы
  целтаб mp[0:1] | чётный и нечётный максимумы, кратные p
  цел x,y | пара чисел для ответа

  m[0] := 0; m[1] := 0
  mp[0] := 0; mp[1] := 0
  ввод N
  нц N раз
    ввод a
    d := mod(a,2)
    если mod(a,p) = 0 и a >= mp[d] | нужно нестрогое сравнение!
      то если mp[d] > m[d] | допустимо нестрогое сравнение
           то  m[d] := mp[d]
         все
         mp[d] := a
      иначе если a > m[d] | допустимо нестрогое сравнение
           то m[d] := a
         все
    все
  кц
  x := 0; y := 0
  если mp[0] > 0 и m[0] > 0
    то x := mp[0]; y := m[0]
  все
  если mp[1] > 0 и m[1] > 0 и mp[1] + m[1] > x + y
    то x := mp[1]; y := m[1]
  все
  вывод x, ' ', y
кон
```

В приведённом решении вместо массивов *m* и *mp* можно использовать две пары простых переменных. В этом случае программа получится более громоздкой (придётся отдельно разбирать случаи чётного и нечётного чисел в последовательности и дублировать аналогичные фрагменты программы), но останется эффективной. Ниже приводится реализующая такой алгоритм программа на языке Python 3

Пример 2. Программа на языке Python 3. Программа эффективна по времени и памяти

```
p = 19
m0 = m1 = mp0 = mp1 = 0
N = int(input())
for i in range(N):
    a = int(input())
    if a % 2 == 0:
        if a % p == 0 and a >= mp0:
            if mp0 > m0: m0 = mp0
            mp0 = a
        elif a > m0: m0 = a
    else:
        if a % p == 0 and a >= mp1:
            if mp1 > m1: m1 = mp1
            mp1 = a
        elif a > m1: m1 = a
x = y = 0
if mp0 > 0 and m0 > 0:
    x = mp0; y = m0
if mp1 > 0 and m1 > 0 and mp1 + m1 > x + y:
    x = mp1; y = m1
print(x,y)
```

Ещё один путь решения – записать всю последовательность в массив и анализировать её в несколько проходов. Ниже приводится реализующая такой алгоритм программа на языке C++. В этой программе массив с исходными данными обрабатывается два раза: на первом проходе находятся индексы максимального чётного и нечётного элементов, кратных p, на втором проходе – общие чётный и нечётный максимумы. При этом элементы, выделенные как кратные при первом проходе, во время второго прохода из сравнения исключаются. Такая программа эффективна по времени (несмотря на повторную обработку массива, общее время работы пропорционально N), но неэффективна по памяти. Максимальная оценка за такую программу при отсутствии в ней синтаксических и содержательных ошибок – 3 балла

Пример 3. Правильная программа на языке C++, эффективная только по времени

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

int main() {
  const int p = 19;  // делитель
  int N; cin >> N;  // количество элементов
  int a[N];       // элементы последовательности
  for (int i = 0; i < N; ++i) cin >> a[i];
  int imp0 = -1, imp1 = -1; //индексы максимумов, кратных p
  for (int i = 0; i < N; ++i) {
    if (a[i] % p == 0) {
      if (a[i] % 2 == 0) {
        if (imp0 == -1 || a[i] > a[imp0]) imp0 = i;
      }
      else {
        if (imp1 == -1 || a[i] > a[imp1]) imp1 = i;
      }
    }
  }
  int im0 = -1, im1 = -1; // индексы общих максимумов
  for (int i = 0; i < N; ++i) {
    if (i != imp0 && i != imp1) {
      if (a[i] % 2 == 0) {
        if (im0 == -1 || a[i] > a[im0]) im0 = i;
      }
      else {
        if (im1 == -1 || a[i] > a[im1]) im1 = i;
      }
    }
  }
  int x = 0, y = 0; // пара чисел для ответа
  if (imp0 != -1 && im0 != -1) {
    x = a[imp0]; y = a[im0];
  }
  if (imp1 != -1 && im1 != -1 && a[imp1] + a[im1] > x + y) {
    x = a[imp1]; y = a[im1];
  }
  cout << x << ' ' << y << endl;
  return 0;
}
```

Возможно также «лобовое» решение: запишем все исходные числа в массив, переберём все возможные пары и выберем подходящую. Такое решение не является эффективным ни по памяти (требуемая память зависит от размера исходных данных), ни по времени (количество возможных пар, а значит, количество действий и время счёта с ростом количества исходных элементов растут квадратично). Подобная программа оценивается не выше 2 баллов.

Ниже приведена реализующая описанный выше алгоритм программа на языке Паскаль (использована версия PascalABC)

Пример 4. Правильная, но неэффективная программа на языке Паскаль

```
const p = 19;
var
    N: integer;     {количество чисел}
    a: array [1..10000] of integer;  {исходные данные}
    x1, x2: integer;     {ответ – пара чисел}
    i,j: integer;

begin
    readln(N);
    for i := 1 to N do readln(a[i]);
    x1 := 0; x2 := 0;
    for i := 1 to N - 1 do begin
        for j := i + 1 to N do begin
            if ((a[i] - a[j]) mod 2 = 0) and
               ((a[i] mod p = 0) or (a[j] mod p = 0)) and
               (a[i] + a[j] > x1 + x2)
            then begin
                x1 := a[i]; x[2] := a[j]
            end
        end
    end;
    writeln(x1, ' ', x2)
end.
```

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. Используемая память не зависит от количества прочитанных чисел N, 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, недопустимая 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 в целом работает правильно для любых входных данных произвольного размера. Время работы пропорционально количеству введённых чисел N.	3

Количество синтаксических ошибок («описок»), указанных в критериях на 4 балла, – не более пяти. Допускается наличие не более одной содержательной ошибки следующих видов: 1) ошибка при вводе данных (не считывается значение N или неверно организован ввод последовательности); 2) ошибка при инициализации или отсутствие инициализации там, где она необходима; 3) используется неверный тип данных; 4) использована одна переменная (константа) вместо другой; 5) используется один знак операции вместо другого; 6) отсутствует вывод ответа, или выводится не то значение (например, выводится сумма вместо пары чисел); 7) неверная работа с массивом, в том числе выход за границы массива; 8) пропущены или неверно расставлены операторные скобки (при использовании языков с операторными скобками); 9) учитываются пары из двух одинаковых чисел, если это число встречается в последовательности только один раз, или не учитываются пары из одинаковых чисел, встречающихся в последовательности более одного раза; 10) используется строгое неравенство там, где необходимо нестрогое, или наоборот. 3 балла также ставится за программу без содержательных ошибок, в которой используемая память зависит от количества прочитанных чисел (например, входные данные запоминаются в массиве или другой аналогичной структуре данных).	
Не выполнены условия, позволяющие поставить 3 или 4 балла, при этом программа работает в целом верно и эффективно по времени. Допускается наличие до трёх содержательных ошибок, описанных в критериях на 3 балла, и до девяти синтаксических ошибок, описанных в критериях на 4 балла. ИЛИ Представлено корректное переборное решение, в котором все исходные данные сохраняются в массиве (или другой аналогичной структуре) и рассматриваются все возможные пары. При этом не допускаются содержательные логические ошибки, например выход индексов за границы массива, рассмотрение сумм вида $a[i] + a[i]$.	2

Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. При этом программа описывает в целом правильный алгоритм, содержит как минимум два элемента из следующего списка, возможно, реализованных с ошибками: 1) рассматриваются пары с чётной разностью; 2) проверяется делимость на *p*; 3) рассматриваются пары с максимальной суммой	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл	*4*

Ответы к заданиям

№ задания	Ответ
1	9
2	wzyx
3	46
4	2
5	110
6	32
7	19
8	240
9	12
10	256
11	1421731
12	192
13	14
14	98
15	8
16	34
17	11
18	91
19	155
20	502
21	61
22	28
23	64

Критерии оценивания заданий с развёрнутым ответом

24 Требовалось написать программу, которая получает на вход натуральное число N, не превосходящее 10^9, и выводит число, равное количеству цифр 4 в десятичной записи числа N. Программист написал программу неправильно. Ниже эта написанная им программа для Вашего удобства приведена на пяти языках программирования.

Бейсик

```
DIM N AS LONG
DIM R, d AS INTEGER
INPUT N
R = 0
WHILE N > 0
  d = N MOD 10
  IF d <> 4 THEN
    R = R + d
  END IF
  N = N \ 10
WEND
PRINT R
END
```

Python

```
N = int(input())
R = 0
while N > 0:
    d = N % 10
    if d != 4:
       R = R + d
    N = N // 10
print(R)
```

Алгоритмический язык

```
алг
нач
  цел N, R, d
  ввод N
  R := 0
  нц пока N > 0
    d := mod(N, 10)
    если d <> 4 то
      R := R + d
    все
    N := div(N, 10)
  кц
  вывод R
кон
```

Паскаль

```
var N: longint;
  R, d: integer;
begin
  readln(N);
  R := 0;
  while N > 0 do begin
    d := N mod 10;
    if d <> 4 then
      R := R + d;
    N := N div 10;
  end;
  writeln(R);
end.
```


C++

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

int main()
{
  long int N;
  int R, d;
  cin >> N;
  R = 0;
  while (N > 0) {
    d = N % 10;
    if (d != 4) {
      R = R + d;
    }
    N = N / 10;
  }
  cout << R << endl;
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 241.
2. Приведите пример входного числа N, при котором приведённая программа, несмотря на ошибки, выдаёт верный 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 нужно исправить не более двух строк 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 любом из четырёх других языков программирования.
1. Программа выведет число 3.
Примечание для эксперта. Программа выводит сумму цифр, отличных от 4.
2. Программа выдаёт правильный ответ, например, при $N = 244$.
3. В программе есть две ошибки.
Первая ошибка: неверная проверка условия увеличения счетчика – переменной *R*.
Строка с ошибкой:
`if d <> 4 then`
Верное исправление:
`if d = 4 then`
Вторая ошибка: неверно увеличивается значение переменной *R*.
Строка с ошибкой:
`R := R + d;`
Верное исправление:
`R := R + 1;`	

Указания по оцениванию	**Баллы**
Обратите внимание! В задаче требовалось выполнить **четыре** действия: 1) указать, что выведет программа при конкретном входном числе; 2) указать пример входного числа, при котором программа выдаёт верный 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 из четырёх необходимых действий	1
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 целые значения от 0 до 10 000 включительно. Опишите на одном из языков программирования алгоритм, который находит количество элементов массива, больших 100 и при этом не кратных 4, а затем заменяет каждый такой элемент на число, равное найденному количеству. 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:

141
256
92
148
511
4

программа должна вывести следующий массив:

2
256
92
148
2
4

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 = 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 := 0;
for i := 1 to N do
  if (a[i] > 100) and (a[i] mod 4 <> 0) then
    k := k + 1;
for i := 1 to N do begin
  if (a[i] > 100) and (a[i] mod 4 <> 0) then
    a[i] := k;
  writeln(a[i]);
end;
```

На Алгоритмическом языке

```
k := 0
нц для i от 1 до N
  если a[i] > 100 и mod(a[i], 4) <> 0
  то
    k := k + 1
  все
кц
нц для i от 1 до N
  если a[i] > 100 и mod(a[i], 4) <> 0
  то
    a[i] := k
  все
  вывод a[i], нс
кц
```

На языке Бейсик

```
K = 0
FOR I = 1 TO N
  IF A(I) > 100 AND A(I) MOD 4 <> 0 THEN
    K = K + 1
  END IF
NEXT I
FOR I = 1 TO N
  IF A(I) > 100 AND A(I) MOD 4 <> 0 THEN
    A(I) = K
  END IF
  PRINT A(I)
NEXT I
```

На языке C++

```
k = 0;
for (i = 0; i < N; i++)
  if (a[i] > 100 && a[i] % 4 != 0)
    k++;
for (i = 0; i < N; i++) {
  if (a[i] > 100 && a[i] % 4 != 0)
    a[i] = k;
  cout << a[i] << endl;
}
```

На языке Python	

`k = 0`
`for i in range(0, n):`
`  if (a[i] > 100 and a[i] % 4 != 0):`
`  k = k + 1`
`for i in range(0, n):`
`  if (a[i] > 100 and 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 (в том числе без разделителей)	
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 количество найденных элементов; 3) неверно осуществляется проверка делимости на 4; 4) проверяется делимость на 4 не элемента массива, а его индекса; 5) неверно выбрана операция отношения для сравнения с границей диапазона; 6) сравнение с границей диапазона производится для индекса элемента массива, а не для его значения; 7) неверно составлено логическое условие (например, используется or вместо and); 8) не вычисляется или неверно вычисляется количество найденных элементов;	1

9) исходный массив не изменяется; 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	
Ошибок, перечисленных в п. 1–13, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии в явном или неявном виде цикла подсчёта количества нужных элементов)	0
Максимальный балл	*2*

26 Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один** или **четыре** камня либо увеличить количество камней в куче **в пять раз**. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16, 19 или 75 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится не менее 63.

Победителем считается игрок, сделавший последний ход, т.е. первым получивший кучу, в которой будет 63 или больше камней.

В начальный момент в куче было S камней; $1 \le S \le 62$.

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. Во всех случаях обосновывайте свой ответ.

Задание 1

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

б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

Задание 2

Укажите два таких значения *S*, при которых 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 каждого 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). На рёбрах дерева указывайте, кто делает ход; в узлах – количество камней в куче.

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, если *S* = 13, … 62.

б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче *S* = 12 камней. Тогда после первого хода Пети в куче будет 13, 16 или 60 камней. Во всех случаях Ваня увеличивает количество камней в 5 раз и выигрывает за один ход.

Замечание для проверяющего. В задаче не требуется указать **все** выигрышные стратегии. Если в работе выпускника, как в приведённом примере, просто указано, что Ваня всегда увеличивает в 5 раз количество камней, – это не ошибка.

Задание 2

Возможные значения *S*: 8, 11. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 12 камней. Эта позиция разобрана в п. 1б. В ней игрок, который будет ходить (теперь это Ваня), выиграть не может, а его противник (т.е. Петя) следующим ходом выиграет.

Задание 3

Возможные значения *S*: 7, 10.

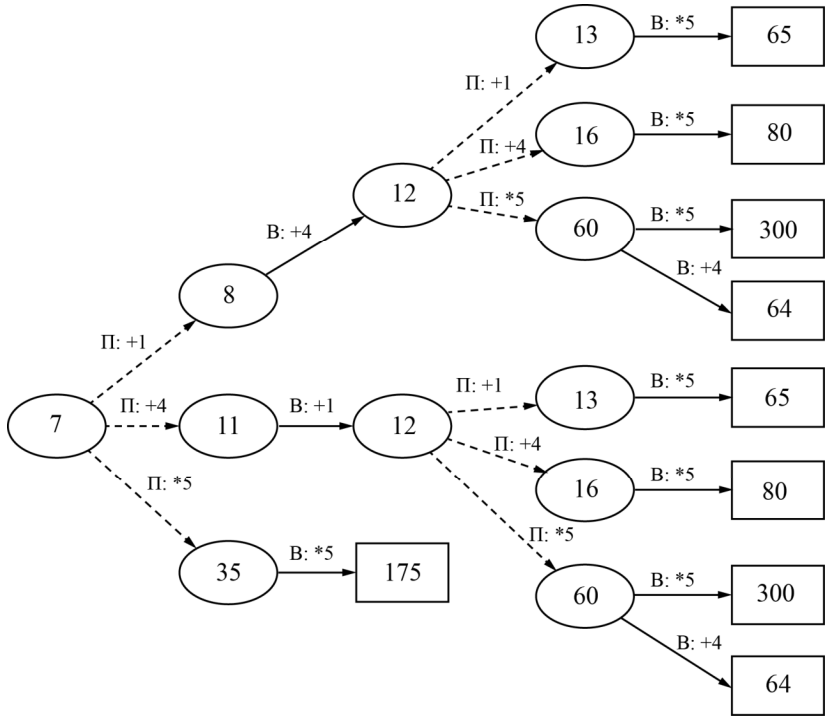
Например, для *S* = 7 после первого хода Пети в куче будет 8, 11 или

35 камней. Если в куче станет 35 камней, Ваня увеличит количество камней в 5 раз и выиграет первым ходом. Ситуация, когда в куче 8 или 11 камней, разобрана в п. 2. В этой ситуации игрок, который будет ходить (теперь это Ваня), выигрывает своим вторым ходом.

В таблице изображено дерево возможных партий (и только их) при описанной стратегии Вани для значения *S* = 7. При выбранной стратегии на последнем ходе Ваня увеличивает в 5 раз количество камней, хотя возможны и другие выигрышные стратегии. Для второго возможного значения дерево строится аналогично. Заключительные позиции (в них выигрывает Ваня) подчёркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

Примечание для проверяющего. Здесь для полноты картины указаны два возможных значения *S*. По условию задачи достаточно указать одно из них

	Положения после очередных ходов			
Исходное положение	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)
7	7 +1 = **8**	8 + 4 = **12**	12 + 1 = **13**	13 * 5 = **65**
			12 + 4 = **16**	16 * 5 = **80**
			12 * 5 = **60**	60 * 5 = **300**
				60 + 4 = **64**
	7 + 4 = **11**	11 + 1 = **12**	12 + 1 = **13**	13 * 5 = **65**
			12 + 4 = **16**	16 * 5 = **80**
			12 * 5 = **60**	60 * 5 = **300**
				60 + 4 = **64**
	7 * 5 = **35**	35 * 5 = **175**		



Дерево всех партий, возможных при Ваниной стратегии.
Прямоугольником обозначены позиции, в которых партия заканчивается

Замечание для проверяющего. На рисунке для наглядности ходы Пети показаны пунктиром, а заключительные позиции выделены прямоугольником. Это не является обязательным для экзаменуемых. Также не является ошибкой указание только одного заключительного хода Вани в ситуации, когда у него есть два заключительных выигрышных хода

Указания по оцениванию	**Баллы**
В задаче от выпускника 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). Если хотя бы один из этих пунктов не выполнен или выполнен с ошибкой (кроме оговоренных выше ошибок), задание считается невыполненным. Задание 2 выполнено, если правильно указаны обе позиции, выигрышные для Пети, и описана соответствующая стратегия Пети – так, как это написано 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 2 или 3 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 Дана последовательность N целых положительных чисел. Рассматриваются все пары элементов последовательности, разность которых чётна, и в этих парах, по крайней мере, одно из чисел пары делится на 17. Порядок элементов в паре неважен. Среди всех таких пар нужно найти и вывести пару с максимальной суммой элементов. Если одинаковую максимальную сумму имеет несколько пар, можно вывести любую из них. Если подходящих пар в последовательности нет, нужно вывести два нуля.

Описание входных и выходных данных

В первой строке входных данных задаётся количество чисел N ($2 \le N \le 10\,000$). В каждой из последующих N строк записано одно натуральное число, не превышающее 10 000.

Пример входных данных:

```
5
34
12
51
52
51
```

Пример выходных данных для приведённого выше примера входных данных:
```
51 51
```

Пояснение. Из данных пяти чисел можно составить три различные пары, удовлетворяющие условию: (34, 12), (34, 52), (51, 51). Наибольшая сумма получается в паре (51, 51). Эта пара допустима, так как число 51 встречается в исходной последовательности дважды.

Напишите эффективную по времени и памяти программу для решения этой 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 Кбайт 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 или только по памяти, – 3 балла.

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

Вы можете сдать **одну** или **две** программы решения задачи. Если Вы сдадите две программы, каждая из них будет оцениваться независимо от другой, итоговой станет **бóльшая** из двух оценок.
Перед текстом программы кратко опишите алгоритм решения. Укажите использованный язык программирования и его версию.

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

Разность двух чисел чётна, если эти числа имеют одинаковую чётность.

Будем вводить элементы последовательности по одному и хранить четыре элемента: максимальные чётный m_0 и нечётный m_1 элементы всей последовательности и максимальные чётный и нечётный элементы, кратные $p = 17$ (m_{p0} и m_{p1} соответственно). При этом у чисел, кратных p, будет «приоритет»: если вновь найденный общий максимум последовательности (с учётом чётности) кратен p, это число будет записано в качестве максимума среди кратных p (m_{p0} или m_{p1}). В общий максимум (m_0 или m_1) оно сможет перейти, только если среди кратных p (m_{p0} или m_{p1}) появится большее или такое же значение.

После ввода всех элементов нужно найти значения $m_{p0} + m_0$, $m_{p1} + m_1$ и выбрать пару с большей суммой. При этом необходимо убедиться, что эти максимумы действительно выбраны из последовательности, а не записаны при инициализации нулевыми или отрицательными значениями.

Ниже приведены программы на языках Pascal (использована версия PascalABC) и Алгоритмическом, реализующие этот алгоритм

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

```
const p = 17;
var
    N: integer;      {количество чисел}
    a: integer;      {очередное число}
    d: integer;      {чётность очередного числа}

    m: array [0..1] of integer;  {чётный и нечётный максимумы}
    mp: array [0..1] of integer;
                      {чётный и нечётный максимумы, кратные p}
    x, y: integer;     {ответ – пара чисел}
    i: integer;

begin
   m[0] := 0; m[1] := 0;
   mp[0] := 0; mp[1] := 0;
   x := 0; y := 0;
   readln(N);
   for i := 1 to N do
     begin
       readln(a);
         d := a mod 2;
         if (a mod p = 0)and(a >= mp[d]) then
                              {нестрогое сравнение!}
           begin
             if mp[d] > m[d] then {допустимо нестрогое сравнение}
                 m[d] := mp[d];
             mp[d] := a
           end
           else if a > m[d] then {допустимо нестрогое сравнение}
             m[d] := a
     end;
   if (mp[0] > 0) and (m[0] > 0) then
       x := mp[0]; y := m[0];
   if (mp[1] > 0) and (m[1] > 0) and (mp[1] + m[1] > x + y) then
       x := mp[1]; y := m[1];
   writeln(x, ' ', y)
end.
```

Пример 1а. Программа на Алгоритмическом языке. Программа эффективна по времени и памяти

```
алг задача
нач
  цел p = 17| делитель из условия задачи
  цел N | количество чисел
  цел a | очередное число
  цел d | чётность очередного числа
  целтаб m[0:1] |чётный и нечётный максимумы
  целтаб mp[0:1] | чётный и нечётный максимумы, кратные p
  цел x,y | пара чисел для ответа

  m[0] := 0; m[1] := 0
  mp[0] := 0; mp[1] := 0
  ввод N
  нц N раз
    ввод a
    d := mod(a,2)
    если mod(a,p) = 0 и a >= mp[d] | нужно нестрогое сравнение!
      то если mp[d] > m[d] | допустимо нестрогое сравнение
           то m[d] := mp[d]
         все
         mp[d] := a
      иначе если a > m[d] | допустимо нестрогое сравнение
           то m[d] := a
         все
    все
  кц
  x := 0; y := 0
  если mp[0] > 0 и m[0] > 0
    то x := mp[0]; y := m[0]
  все
  если mp[1] > 0 и m[1] > 0 и mp[1] + m[1] > x + y
    то x := mp[1]; y := m[1]
  все
  вывод x, ' ', y
кон
```

В приведённом решении вместо массивов *m* и *mp* можно использовать две пары простых переменных. В этом случае программа получится более громоздкой (придётся отдельно разбирать случаи чётного и нечётного чисел в последовательности и дублировать аналогичные фрагменты программы), но останется эффективной. Ниже приводится реализующая такой алгоритм программа на языке Python 3

Пример 2. Программа на языке Python 3. Программа эффективна по времени и памяти

```
p = 17
m0 = m1 = mp0 = mp1 = 0
N = int(input())
for i in range(N):
    a = int(input())
    if a % 2 == 0:
        if a % p == 0 and a >= mp0:
            if mp0 > m0: m0 = mp0
            mp0 = a
        elif a > m0: m0 = a
    else:
        if a % p == 0 and a >= mp1:
            if mp1 > m1: m1 = mp1
            mp1 = a
        elif a > m1: m1 = a
x = y = 0
if mp0 > 0 and m0 > 0:
    x = mp0; y = m0
if mp1 > 0 and m1 > 0 and mp1 + m1 > x + y:
    x = mp1; y = m1
print(x,y)
```

Ещё один путь решения – записать всю последовательность в массив и анализировать её в несколько проходов. Ниже приводится реализующая такой алгоритм программа на языке C++. В этой программе массив с исходными данными обрабатывается два раза: на первом проходе находятся индексы максимального чётного и нечётного элементов, кратных *p*, на втором проходе – общие чётный и нечётный максимумы. При этом элементы, выделенные как кратные при первом проходе, во время второго прохода из сравнения исключаются. Такая программа эффективна по времени (несмотря на повторную обработку массива, общее время работы пропорционально *N*), но неэффективна по памяти. Максимальная оценка за такую программу при отсутствии в ней синтаксических и содержательных ошибок – 3 балла

Пример 3. Правильная программа на языке C++, эффективная только по времени

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

int main() {
  const int p = 17;  // делитель
  int N; cin >> N;  // количество элементов
  int a[N];       // элементы последовательности
  for (int i = 0; i < N; ++i) cin >> a[i];
  int imp0 = -1, imp1 = -1; //индексы максимумов, кратных p
  for (int i = 0; i < N; ++i) {
    if (a[i] % p == 0) {
      if (a[i] % 2 == 0) {
        if (imp0 == -1 || a[i] > a[imp0]) imp0 = i;
      }
      else {
        if (imp1 == -1 || a[i] > a[imp1]) imp1 = i;
      }
    }
  }
  int im0 = -1, im1 = -1; // индексы общих максимумов
  for (int i = 0; i < N; ++i) {
    if (i != imp0 && i != imp1) {
      if (a[i] % 2 == 0) {
        if (im0 == -1 || a[i] > a[im0]) im0 = i;
      }
      else {
        if (im1 == -1 || a[i] > a[im1]) im1 = i;
      }
    }
  }
  int x = 0, y = 0; // пара чисел для ответа
  if (imp0 != -1 && im0 != -1) {
    x = a[imp0]; y = a[im0];
  }
  if (imp1 != -1 && im1 != -1 && a[imp1] + a[im1] > x + y) {
    x = a[imp1]; y = a[im1];
  }
  cout << x << ' ' << y << endl;
  return 0;
}
```

Возможно также «лобовое» решение: запишем все исходные числа в массив, переберём все возможные пары и выберем подходящую. Такое решение не является эффективным ни по памяти (требуемая память зависит от размера исходных данных), ни по времени (количество возможных пар, а значит, количество действий и время счёта с ростом количества исходных элементов растут квадратично). Подобная программа оценивается не выше 2 баллов.

Ниже приведена реализующая описанный выше алгоритм программа на языке Паскаль (использована версия PascalABC)

Пример 4. Правильная, но неэффективная программа на языке Паскаль

```
const p = 17;
var
    N: integer;     {количество чисел}
    a: array [1..10000] of integer;  {исходные данные}
    x1, x2: integer;     {ответ – пара чисел}
    i,j: integer;

begin
    readln(N);
    for i := 1 to N do readln(a[i]);
    x1 := 0; x2 := 0;
    for i := 1 to N - 1 do begin
        for j := i + 1 to N do begin
            if ((a[i] - a[j]) mod 2 = 0) and
               ((a[i] mod p = 0) or (a[j] mod p = 0)) and
               (a[i] + a[j] > x1 + x2)
            then begin
                x1 := a[i]; x[2] := a[j]
            end
        end
    end;
    writeln(x1, ' ', x2)
end.
```

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. Используемая память не зависит от количества прочитанных чисел *N*, 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, недопустимая 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 в целом работает правильно для любых входных данных произвольного размера. Время работы пропорционально количеству введённых чисел *N*.	3

Количество синтаксических ошибок («описок»), указанных в критериях на 4 балла, – не более пяти. Допускается наличие не более одной содержательной ошибки следующих видов: 1) ошибка при вводе данных (не считывается значение N или неверно организован ввод последовательности); 2) ошибка при инициализации или отсутствие инициализации там, где она необходима; 3) используется неверный тип данных; 4) использована одна переменная (константа) вместо другой; 5) используется один знак операции вместо другого; 6) отсутствует вывод ответа, или выводится не то значение (например, выводится сумма вместо пары чисел); 7) неверная работа с массивом, в том числе выход за границы массива; 8) пропущены или неверно расставлены операторные скобки (при использовании языков с операторными скобками); 9) учитываются пары из двух одинаковых чисел, если это число встречается в последовательности только один раз, или не учитываются пары из одинаковых чисел, встречающихся в последовательности более одного раза; 10) используется строгое неравенство там, где необходимо нестрогое, или наоборот. 3 балла также ставится за программу без содержательных ошибок, в которой используемая память зависит от количества прочитанных чисел (например, входные данные запоминаются в массиве или другой аналогичной структуре данных)	
Не выполнены условия, позволяющие поставить 3 или 4 балла, при этом программа работает в целом верно и эффективно по времени. Допускается наличие до трёх содержательных ошибок, описанных в критериях на 3 балла, и до девяти синтаксических ошибок, описанных в критериях на 4 балла. ИЛИ Представлено корректное переборное решение, в котором все исходные данные сохраняются в массиве (или другой аналогичной структуре) и рассматриваются все возможные пары. При этом не допускаются содержательные логические ошибки, например выход индексов за границы массива, рассмотрение сумм вида $a[i] + a[i]$.	2
Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. При этом программа описывает в целом правильный алгоритм, содержит как минимум два элемента из следующего списка, возможно, реализованных с ошибками: 1) рассматриваются пары с чётной разностью; 2) проверяется делимость на p; 3) рассматриваются пары с максимальной суммой	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл	*4*