ВАРИАНТ 530
Критерии оценивания заданий с развёрнутым ответом

24 На обработку поступает натуральное число, не превышающее 10^9. Нужно написать программу, которая выводит на экран количество цифр этого числа, кратных 5. Если в числе нет цифр, кратных 5, требуется на экран вывести «NO». Программист написал программу неправильно. Ниже эта программа для Вашего удобства приведена на пяти языках программирования.

Напоминание: 0 делится на любое натуральное число.

Бейсик

```
DIM N, DIGIT, COUNT AS LONG
INPUT N
COUNT = 1
WHILE N > 0
  DIGIT = N MOD 10
  IF DIGIT MOD 5 = 0 THEN
    COUNT = COUNT + DIGIT
  END IF
  N = N \ 10
WEND
IF COUNT = 0 THEN
  PRINT "NO"
ELSE
  PRINT COUNT
END IF
```

Python

```
N = int(input())
count = 1
while N > 0:
  digit = N % 10
  if digit % 5 == 0:
    count = count + digit
  N = N // 10
if count == 0:
  print("NO")
else:
  print(count)
```

Алгоритмический язык

```
алг
нач
  цел N, digit, count
  ввод N
  count := 1
  нц пока N > 0
    digit := mod(N,10)
    если mod(digit,5) = 0 то
      count := count + digit
    все
    N := div(N,10)
  кц
  если count = 0 то
    вывод "NO"
  иначе
    вывод count
  все
кон
```

Паскаль

```
var N, digit, count: longint;
begin
  readln(N);
  count := 1;
  while N > 0 do
  begin
    digit := N mod 10;
    if digit mod 5 = 0  then
      count := count + digit;
    N := N div 10;
  end;
  if count = 0 then
    writeln('NO')
  else
    writeln(count)
end.
```

Си

```
#include <stdio.h>
int main()
{
  int N, digit, count;
  scanf("%d", &N);
  count = 1;
  while (N > 0)
  {
    digit = N % 10;
    if (digit % 5 == 0)
      count = count + digit;
    N = N / 10;
  }
  if (count == 0)
    printf("NO");
  else
    printf("%d",count);
  return 0;
}
```

Последовательно выполните следующее.

1. Напишите, что выведет эта программа при вводе числа 164.
2. Приведите пример такого трёхзначного числа, при вводе которого программа выдаёт верный ответ.
3. Найдите все ошибки в этой программе (их может быть одна или несколько). Известно, что каждая ошибка затрагивает только одну строку и может быть исправлена без изменения других строк. Для каждой ошибки:
   1) выпишите строку, в которой сделана ошибка;
   2) укажите, как исправить ошибку, т.е. приведите правильный вариант строки.

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

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

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

Решение использует запись программы на Паскале. Допускается использование программы на любом из четырёх других языков.
1. Программа выведет число 1.
2. Программа выдаёт правильный ответ, например, для числа 160.
Замечание для проверяющего. Программа работает неправильно из-за неверного задания начального значения счётчика и неверного увеличения счётчика. Соответственно, программа будет работать верно, если в числе есть ровно один значащий 0 и нет других цифр, кратных 5.
3. В программе есть две ошибки.
Первая ошибка. Неверное начальное значение счётчика.
Строка с ошибкой:
`count := 1;`
Верное исправление:
`count := 0;`
Вторая ошибка. Неверное изменение счётчика.
Строка с ошибкой:
`count := count + digit;`
Верное исправление:
`count := count + 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 необходимых действия из четырёх	1
Не выполнены условия, позволяющие поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

25 Дан целочисленный массив из 20 элементов. Элементы массива могут принимать целые значения от –10 000 до 10 000 включительно. Опишите на естественном языке или на одном из языков программирования алгоритм, позволяющий найти и вывести количество пар элементов массива, в которых хотя бы одно число делится на 13. В данной задаче под парой подразумевается два подряд идущих элемента массива.
Например, для массива из пяти элементов: 6; 2; 13; –26; 14 – ответ: 3.

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

Бейсик

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

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

END
```

Python

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

Алгоритмический язык

```
алг
нач
    цел N = 20
    целтаб a[1:N]
    цел i, j, k
    нц для i от 1 до N
        ввод a[i]
    кц
    ...

кон
```

Паскаль

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

end.
```

Си

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

Естественный язык

Объявляем массив *A* из 20 элементов.
Объявляем целочисленные переменные *I, J, K*.
В цикле от 1 до 20 вводим элементы массива *A* с 1-го по 20-й.
…

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

Содержание верного ответа и указания по оцениванию
(допускаются иные формулировки решений, приводящие к правильному результату)

На языке Паскаль

```
k := 0;
for i := 1 to N-1 do
    if (a[i] mod 13=0) or (a[i+1] mod 13=0) then
        inc(k);
writeln(k);
```

На алгоритмическом языке

```
k := 0;
нц для i от 1 до N-1
    если mod(a[i],13)=0 или mod(a[i+1],13)=0
    то
        k := k+1
    все
кц
вывод k
```

На языке Бейсик

```
K = 0
FOR I = 1 TO N-1
    IF (A(I) MOD 13 = 0) OR (A(I + 1) MOD 13 = 0) THEN
        K = K+1
    END IF
NEXT I
PRINT K
```

На языке Си

```
k = 0;
for (i = 0; i<N-1; i++)
    if (a[i]%13 == 0 || a[i+1]%13 == 0)
        k++;
printf("%d", k);
```

На языке Python

```
k = 0
for i in range(0, n - 1):
    if (a[i] % 13 == 0 or a[i + 1] % 13 == 0):
        k += 1
print(k)
```

На естественном языке	
Записываем в переменную *K* начальное значение, равное 0. В цикле от первого элемента до предпоследнего находим остаток от деления текущего и следующего элемента массива на 13. Если хотя бы один остаток равен 0, увеличиваем переменную *K* на единицу. После завершения цикла выводим значение переменной *K*	
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, перечисленных 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 естественном языке. Использование нетипизированных или необъявленных переменных возможно только в случае, если это допускается языком программирования; при этом количество переменных и их идентификаторы должны соответствовать условию задачи	

Предложен правильный алгоритм, выдающий в качестве результата верное значение	2
Не выполнены условия, позволяющие поставить 2 балла. Предложено в целом верное решение, содержащее не более одной ошибки из числа следующих: 1) в цикле происходит выход за границу массива (например, используется цикл от 1 до *N*); 2) не инициализируется или неверно инициализируется счётчик количества найденных пар; 3) счётчик количества пар в цикле не изменяется или изменяется неверно; 4) неверно проверяется делимость на 13; 5) на делимость проверяются не сами элементы, а их индексы; 6) при проверке выполнения условия для пары элементов используются неверные индексы; 7) в сложном логическом условии простые проверки верны, но условие в целом построено неверно (например, перепутаны операции «И» и «ИЛИ», неверно расставлены скобки в логическом выражении); 8) отсутствует вывод ответа; 9) используется переменная, не объявленная в разделе описания переменных; 10) не указано или неверно указано условие завершения цикла; 11) индексная переменная в цикле не меняется (например, в цикле while) или меняется неверно; 12) неверно расставлены операторные скобки	1
Ошибок, перечисленных в п. 1–12, две или больше, или алгоритм сформулирован неверно (в том числе при отсутствии цикла в явном или неявном виде)	0
Максимальный балл	*2*

26 Два игрока, Петя и Ваня, играют в следующую игру. 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 77. Победителем считается игрок, сделавший последний ход, т.е. первым получивший такую позицию, что в кучах всего будет 77 или больше камней.

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. Например, при начальных позициях (6, 36), (7, 35), (9, 34) выигрышная стратегия есть у Пети. Чтобы выиграть, ему достаточно удвоить количество камней во второй куче.

Задание 1. Для каждой из начальных позиций (6, 35), (8, 34) укажите, кто из игроков имеет выигрышную стратегию. В каждом случае опишите выигрышную стратегию; объясните, почему эта стратегия ведёт к выигрышу, и укажите, какое наибольшее количество ходов может потребоваться победителю для выигрыша при этой стратегии.

Задание 2. Для каждой из начальных позиций (6, 34), (7, 34), (8, 33) укажите, кто из игроков имеет выигрышную стратегию. В каждом случае опишите выигрышную стратегию; объясните, почему эта стратегия ведёт к выигрышу, и укажите, какое наибольшее количество ходов может потребоваться победителю для выигрыша при этой стратегии.

Задание 3. Для начальной позиции (7, 33) укажите, кто из игроков имеет выигрышную стратегию. Опишите выигрышную стратегию; объясните, почему эта стратегия ведёт к выигрышу, и укажите, какое наибольшее количество ходов может потребоваться победителю для выигрыша при этой стратегии. Постройте дерево всех партий, возможных при указанной Вами выигрышной стратегии. Представьте дерево в виде рисунка или таблицы.

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

Задание 1. В начальных позициях (6, 35), (8, 34) выигрышная стратегия есть у Вани. При начальной позиции (6, 35) после первого хода Пети может получиться одна из следующих четырёх позиций: (7, 35), (12, 35), (6, 36), (6, 70). Каждая из этих позиций содержит менее 77 камней. При этом из любой из этих позиций Ваня может получить позицию, содержащую не менее 77 камней, удвоив количество камней во второй куче. Для позиции (8, 34) после первого хода Пети может получиться одна из следующих четырёх позиций: (9, 34), (16, 34), (8, 35), (8, 68). Каждая из этих позиций содержит менее 77 камней. При этом из любой из этих позиций Ваня может получить позицию, содержащую не менее 77 камней, удвоив количество камней во второй куче. Таким образом, Ваня при любом ходе Пети выигрывает своим первым ходом.

Задание 2. В начальных позициях (6, 34), (7, 34) и (8, 33) выигрышная стратегия есть у Пети. При начальной позиции (6, 34) он должен первым ходом получить позицию (6, 35), из начальных позиций (7, 34) и (8, 33) Петя после первого хода должен получить позицию (8, 34). Позиции (6, 35) и (8, 34) рассмотрены при разборе задания 1. В этих позициях выигрышная стратегия есть у игрока, который будет ходить вторым (теперь это Петя). Эта стратегия описана при разборе задания 1. Таким образом, Петя при любой игре Вани выигрывает своим вторым ходом.

Задание 3. В начальной позиции (7, 33) выигрышная стратегия есть у Вани. После первого хода Пети может возникнуть одна из четырёх позиций: (8, 33), (7, 34), (14, 33) и (7, 66). В позициях (14, 33) и (7, 66) Ваня может выиграть одним ходом, удвоив количество камней во второй куче. Позиции (8, 33) и (7, 34) были рассмотрены при разборе задания 2. В этих позициях у игрока, который должен сделать ход (теперь это Ваня), есть выигрышная стратегия. Эта стратегия описана при разборе задания 2. Таким образом, в зависимости от игры Пети Ваня выигрывает на первом или на втором ходу.

Примечание для эксперта. Последняя фраза в приведённом решении избыточна. Не будет ошибкой, если экзаменуемый просто напишет, например, «При выбранной стратегии партия длится не более двух ходов».

В таблице изображено дерево возможных партий при описанной стратегии Вани. Заключительные позиции (в них выигрывает Ваня) выделены жирным шрифтом.

<table>
<tr><td></td><td colspan="4">Положения после очередных ходов</td></tr>
<tr><td>Исходное положение</td><td>1-й ход Пети (разобраны все ходы, указана полученная позиция)</td><td>1-й ход Вани (только ход по стратегии, указана полученная позиция)</td><td>2-й ход Пети (разобраны все ходы, указана полученная позиция)</td><td>2-й ход Вани (только ход по стратегии, указана полученная позиция)</td></tr>
<tr><td rowspan="13">(7, 33)
Всего: 40</td><td rowspan="4">(7, 33+1) =
(7, 34)
Всего: 41</td><td rowspan="4">(7+1, 34) =
(8, 34)
Всего: 42</td><td>(8+1, 34) =
(9, 34)
Всего: 43</td><td>(9, 34*2) =
(9, 68)
Всего: 77</td></tr>
<tr><td>(8, 34+1) =
(8, 35)
Всего: 43</td><td>(8, 35*2) =
(8, 70)
Всего: 78</td></tr>
<tr><td>(8*2, 34) =
(16, 34)
Всего: 50</td><td>(16, 34*2) =
(16, 68)
Всего: 84</td></tr>
<tr><td>(8, 34*2) =
(8, 68)
Всего: 76</td><td>(8, 68*2) =
(8, 136)
Всего: 144</td></tr>
<tr><td rowspan="4">(7+1, 33) =
(8, 33)
Всего: 41</td><td rowspan="4">(8, 33+1) =
(8, 34)
Всего: 42</td><td>(8+1, 34) =
(9, 34)
Всего: 43</td><td>(9, 34*2) =
(9, 68)
Всего: 77</td></tr>
<tr><td>(8, 34+1) =
(8, 35)
Всего: 43</td><td>(8, 35*2) =
(8, 70)
Всего: 78</td></tr>
<tr><td>(8*2, 34) =
(16, 34)
Всего: 50</td><td>(16, 34*2) =
(16, 68)
Всего: 84</td></tr>
<tr><td>(8, 34*2) =
(8, 68)
Всего: 76</td><td>(8, 68*2) =
(8, 136)
Всего: 144</td></tr>
<tr><td>(7*2, 33) =
(14, 33)
Всего: 47</td><td>(14, 33*2) =
(14, 66)
Всего: 80</td><td></td><td></td></tr>
<tr><td>(7, 33*2) =
(7, 66)
Всего: 73</td><td>(7, 66*2) =
(7, 132)
Всего: 139</td><td></td><td></td></tr>
</table>

Примечание для эксперта. Дерево всех партий может быть также изображено в виде ориентированного графа – так, как показано на рисунке, или другим способом. Например, вершины дерева, соответствующие одной и той же позиции, на рисунке могут быть «склеены». Важно, чтобы множество

полных путей в графе находилось во взаимно однозначном соответствии с множеством партий, возможных при описанной в решении стратегии.

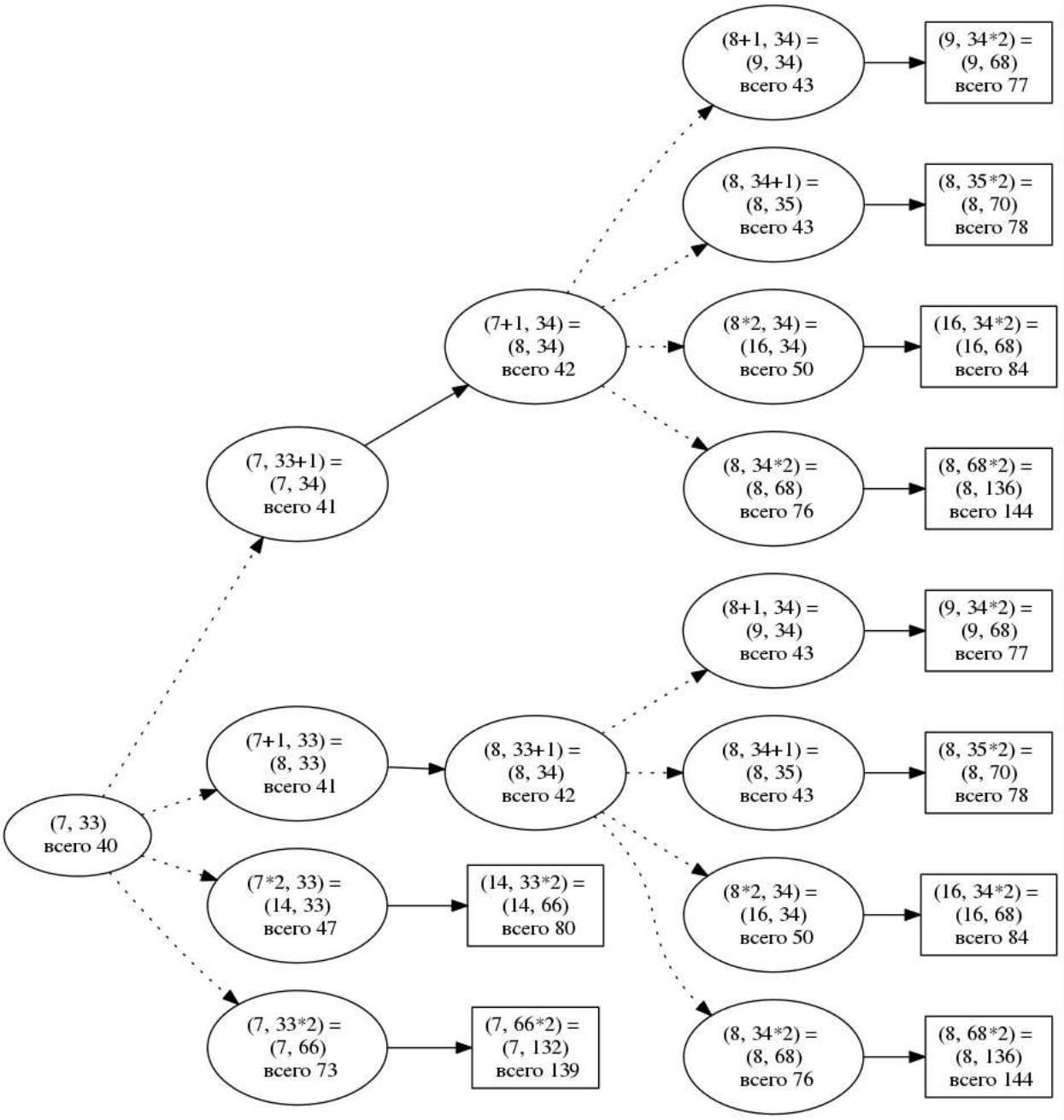

Рис. 1. Дерево всех партий, возможных при описанной стратегии Вани. Ходы Пети показаны пунктирными стрелками, ходы Вани показаны сплошными стрелками. Заключительные позиции обозначены прямоугольником

Примечание для эксперта. В некоторых позициях у Вани есть и другой способ выигрыша: например, в позиции (8, 68) можно добавить один камень в любую кучу. То, что это не указано, не является ошибкой. Экзаменуемый не должен указывать все возможные выигрышные стратегии

Указания по оцениванию	**Баллы**
В задаче от ученика требуется выполнить три задания. 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. Во всех случаях стратегии могут быть описаны так, как это сделано в примере решения, или другим способом	
Выполнены все три задания. Здесь и далее в решениях допускаются арифметические ошибки, которые не искажают сути решения и не приводят к неправильному ответу	3
Не выполнены условия, позволяющие поставить 3 балла, и выполнено хотя бы одно из следующих условий. – Выполнено задание 3. – Выполнены задания 1 и 2	2
Не выполнены условия, позволяющие поставить 2 или 3 балла, и выполнено хотя бы одно из следующих условий. – Выполнено задание 1. – Выполнено задание 2	1
Не выполнено ни одно из условий, позволяющих поставить 1, 2 или 3 балла	0
Максимальный балл	*3*

27 В физической лаборатории проводится долговременный эксперимент по изучению гравитационного поля Земли. По каналу связи каждую минуту в лабораторию передаётся положительное целое число – текущее показание прибора «Сигма 2015». Количество передаваемых чисел в серии известно и не превышает 10 000. Все числа не превышают 1000. Временем, в течение которого происходит передача, можно пренебречь.
Необходимо вычислить «бета-значение» серии показаний прибора – минимальное **чётное** произведение двух показаний, между моментами передачи которых прошло не менее 7 минут. Если получить такое произведение не удаётся, ответ считается равным –1.

Вам предлагается два задания, связанных с этой задачей: задание А и задание Б. Вы можете решать оба задания или одно из них по своему выбору.
Итоговая оценка выставляется как ***максимальная*** *из оценок за задания А и Б. Если решение одного из заданий не представлено, то считается, что оценка за это задание – 0 баллов.*
Задание Б является усложнённым вариантом задания А, оно содержит дополнительные требования к программе.

А. Напишите на любом языке программирования программу для решения поставленной задачи, в которой входные данные будут запоминаться в массиве, после чего будут проверены все возможные пары элементов. Перед программой укажите версию языка программирования.
ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ А.
Максимальная оценка за выполнение задания А – 2 балла.
Б. Напишите программу для решения поставленной задачи, которая будет эффективна как по времени, так и по памяти (или хотя бы по одной из этих характеристик).
Программа считается эффективной по времени, если время работы программы пропорционально количеству полученных показаний прибора N, т.е. при увеличении N в k раз время работы программы должно увеличиваться не более чем в k раз.
Программа считается эффективной по памяти, если размер памяти, использованной в программе для хранения данных, не зависит от числа N и не превышает 1 килобайта.
Перед программой укажите версию языка программирования и кратко опишите использованный алгоритм.
ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ Б.
Максимальная оценка за правильную программу, эффективную по времени и по памяти, – 4 балла.
Максимальная оценка за правильную программу, эффективную по времени, но неэффективную по памяти, – 3 балла.

НАПОМИНАЕМ! Не забудьте указать, к какому заданию относится каждая из представленных Вами программ.

Входные данные представлены следующим образом. В первой строке задаётся число N – общее количество показаний прибора. Гарантируется, что $N > 7$. В каждой из следующих N строк задаётся одно положительное целое число – очередное показание прибора.

Пример входных данных:

```
12
12
45
5
3
14
17
23
21
20
19
18
17
```

Программа должна вывести одно число – описанное в условии произведение, либо –1, если получить такое произведение не удаётся.

Пример выходных данных для приведённого выше примера входных данных:

```
54
```

Содержание верного ответа и указания по оцениванию (допускаются иные формулировки ответа, не искажающие его смысла)
Задание Б (решение для задания А приведено ниже, см. программу 4). Чтобы произведение было чётным, хотя бы один сомножитель должен быть чётным, поэтому при поиске подходящих произведений чётные показания прибора можно рассматривать в паре с любыми другими, а нечётные – только с чётными. Для каждого показания с номером k, начиная с $k = 8$, рассмотрим все допустимые по условиям задачи пары, в которых данное показание получено вторым. Минимальное произведение из всех этих пар будет получено, если первым в паре будет взято минимальное подходящее показание среди всех, полученных от начала приёма и до показания с номером $k - 7$. Если очередное показание чётное, минимальное среди предыдущих может быть любым, если нечётное – только чётным. Для получения эффективного по времени решения нужно по мере ввода данных помнить абсолютное минимальное и минимальное чётное показание на каждый момент времени, каждое вновь полученное показание умножать на соответствующий ему минимум, имевшийся на 7 элементов ранее, и выбрать минимальное из всех таких произведений. Поскольку каждое текущее минимальное показание используется после

ввода ещё 7 элементов и после этого становится ненужным, достаточно хранить только 7 последних минимумов. Для этого можно использовать массив из 7 элементов и циклически заполнять его по мере ввода данных. Размер этого массива не зависит от общего количества введённых показаний, поэтому такое решение будет эффективным не только по времени, но и по памяти. Чтобы хранить абсолютный и чётный минимумы, нужно использовать два таких массива.

Ниже приводится пример такой программы, написанной на алгоритмическом языке.

Программа 1. Пример правильной программы на алгоритмическом языке. Программа эффективна по времени и по памяти

```
алг
нач
  цел s = 7        | требуемое расстояние между показаниями
  цел amax = 1001  | больше максимально возможного показания
  цел N
  ввод N
  цел a                    | очередное показание прибора
  целтаб мини[0:s-1]       | текущие минимумы последних s элементов
  целтаб миничет[0:s-1]    | чётные минимумы последних s элементов
  цел i
  | вводим первые s показаний, фиксируем минимумы
  цел ма; ма := amax       | минимальное показание
  цел мчет; мчет := amax   | минимальное чётное показание
  нц для i от 1 до s
    ввод a
    ма := imin(ма, a)
    если mod(a,2) = 0 то мчет := imin(мчет,a) все
    мини[mod(i, s)] := ма
    миничет[mod(i, s)] := мчет
  кц
  цел мп = amax*amax | минимальное значение произведения
  цел п
  нц для i от s+1 до N
    ввод a
    если mod(a,2)=0
      то п := a * мини[mod(i, s)]
      иначе если мчет < amax
        то п := a * миничет[mod(i, s)]
        иначе п := amax*amax;
      все
    все
    мп := imin(мп, п)
    ма := imin(ма, a)
    если mod(a,2) = 0 то мчет := imin(мчет,a) все
    мини[mod(i, s)] := ма
    миничет[mod(i, s)] := мчет
  кц
  если мп = amax*amax то мп:=-1 все
  вывод мп
кон
```

Возможны и другие реализации. Например, вместо циклического заполнения массива можно каждый раз сдвигать его элементы. В приведённом ниже примере хранятся и сдвигаются не минимумы, а исходные значения. Это требует чуть меньше памяти (достаточно одного массива вместо двух), но по времени решение со сдвигами менее эффективно, чем с циклическим заполнением. Однако время работы остаётся пропорциональным N, поэтому максимальная оценка за такое решение тоже составляет 4 балла.

Программа 2. Пример правильной программы на языке Паскаль.
Программа использует сдвиги, но эффективна по времени и по памяти

```
const s = 7; {требуемое расстояние между показаниями}
      amax = 1001; {больше максимально возможного показания}
var
  N: integer;
  a: array[1..s] of integer; {хранение s показаний прибора}
  a_: integer; {ввод очередного показания}
  ma: integer; {минимальное число без s последних}
  me: integer; {минимальное чётное число без s последних}
  mp: integer; {минимальное значение произведения}
  p: integer;
  i, j: integer;
begin
  readln(N);
  {Ввод первых s чисел}
  for i:=1 to s do readln(a[i]);
  {Ввод остальных значений, поиск минимального произведения}
  ma := amax; me := amax;
  mp :=amax*amax;
  for i := s + 1 to N do begin
    readln(a_);
    if a[1] < ma then ma := a[1];
    if (a[1] mod 2 = 0) and (a[1] < me) then me := a[1];
    if a_ mod 2 = 0 then p := a_ * ma
    else if me < amax then p := a_ * me
    else p := amax* amax;
    if (p < mp) then mp := p;
    {сдвигаем элементы вспомогательного массива влево}
    for j := 1 to s - 1 do
      a[j] := a[j + 1];
    a[s] := a_
  end;
  if mp = amax*amax then mp:=-1;
  writeln(mp)
end.
```

Если вместо небольшого массива фиксированного размера (циклического или со сдвигами) хранятся все исходные данные (или все текущие минимумы), программа сохраняет эффективность по времени, но становится неэффективной по памяти, так как требуемая память растёт пропорционально N. Ниже приводится пример такой программы на языке Паскаль. Подобные (и аналогичные по сути) программы оцениваются

не выше 3 баллов.

Программа 3. Пример правильной программы на языке Паскаль.
Программа эффективна по времени, но неэффективна по памяти

```
const s = 7; {требуемое расстояние между показаниями}
      amax = 1001; {больше максимально возможного показания}
var
  N, p, i: integer;
  a: array[1..10000] of integer; {все показания прибора}
  ma: integer; {минимальное число без s последних}
  me: integer; {минимальное чётное число без s последних}
  mp: integer; {минимальное значение произведения}
begin
  readln(N);
  {Ввод всех показаний прибора}
  for i:=1 to N do readln(a[i]);
  ma := amax;
  me := amax;
  mp := amax*amax;
  for i := s + 1 to N do
  begin
    if a[i-s] < ma then ma := a[i-s];
    if (a[i-s] mod 2 = 0) and (a[i-s] < me) then
      me := a[i-s];
    if a[i] mod 2 = 0 then p := a[i] * ma
    else if me < amax then p := a[i] * me
    else p := amax * amax;
    if (p < mp) then mp := p
  end;
  if mp = amax*amax then mp := -1;
  writeln(mp)
end.
```

Возможно также переборное решение, в котором находятся произведения всех возможных пар и из них выбирается минимальное. Ниже (см. программу 4) приведён пример подобного решения. Это (и аналогичные ему) решение неэффективно ни по времени, ни по памяти. Оно является решением задания А, но не является решением задания Б. Оценка за такое решение – 2 балла.

Программа 4. Пример правильной программы на языке Паскаль.
Программа неэффективна ни по времени, ни по памяти

```
const s = 7; {требуемое расстояние между показаниями}
var
  N: integer;
  a: array[1..10000] of integer; {все показания прибора}
  mp: integer; {минимальное значение произведения}
  i, j: integer;
begin
  readln(N);
  {Ввод значений прибора}
  for i:=1 to N do
    readln(a[i]);
  mp := 1000 * 1000 + 1;
  for i := 1 to N-s do begin
    for j := i+s to N do begin
      if (a[i]*a[j] mod 2 = 0) and (a[i]*a[j] < mp)
        then mp := a[i]*a[j]
    end;
  end;
  if mp = 1000 * 1000 + 1 then mp := -1;
  writeln(mp)
end.
```

Указания по оцениванию	**Баллы**
Предварительные замечания. 1. В задаче есть два задания (А и Б). Соответственно, ученик может представить две программы. В каждой из программ должно быть указано, решением какого из заданий она является. Если в работе представлена одна программа, то в ней также должно быть указано, решением какого из заданий она является. 2. Если ученик не указал, к какому заданию относится программа, или можно предположить, что ученик ошибся в идентификации программ, необходимо следовать приведённым ниже инструкциям. Случай 2.1. Ученик представил только одну программу. Следует рассматривать программу как решение задания Б и оценивать её по соответствующим критериям. Случай 2.2. Ученик представил две программы, но указание задания есть только для одной из программ. Следует рассматривать вторую программу как ответ на оставшееся задание. Случай 2.3. Ученик представил две программы; ни для одной из них задание не указано, или в обоих решениях указано одно и то же задание.	

Следует первую (по порядку в представленных учеником материалах) программу рассматривать как ответ на задание А, а вторую – как ответ на задание Б. Случай 2.4. Ученик представил более двух программ. Следует рассматривать только две последние программы и соотносить их с заданиями по правилам 2.1–2.3. Случай 2.5. Решение, представленное в качестве решения задания А, по критериям для задания Б может быть оценено в 3 или 4 балла. При этом решение, представленное в качестве решения задания Б, получило меньшую оценку. Следует считать, что ученик перепутал обозначения заданий и оценивать решение, представленное как решение задания А, по критериям задания Б. НАПОМИНАЕМ! *Итоговый балл за задачу – это* ***больший*** *из баллов, полученных учеником за каждое из двух представленных решений.* *Пояснения для проверяющих.* 1. Задание Б является усложнением задания А. Если в качестве решения задания Б представлено решение задания А, то согласно приведённым ниже критериям его оценка будет такой же, как если бы это решение было представлено в качестве решения задания А. 2. Два задания (и, соответственно, возможность для экзаменуемого представить две программы) дают ученику возможность (при его желании) сначала написать менее сложное и менее эффективное решение (задание А), которое даёт ему право получить 2 балла, а затем приступить к поиску более эффективного решения. 3. Приведённые в п. 2.1–2.5 правила имеют целью избежать снижения оценки из-за того, что ученик перепутал обозначения заданий	
Критерии оценивания задания А	
Программа решает поставленную задачу для любых соответствующих условию входных данных. Например, допускается переборное решение, аналогичное приведённой выше программе 4. Допускается до семи синтаксических и приравненных к ним ошибок (см. критерии оценивания задания Б на 4 балла). Допускается до двух содержательных ошибок, описанных в критериях оценивания задания Б на 3 балла	2

Не выполнены условия, позволяющие поставить 2 балла. Из описания алгоритма или общей структуры программы видно, что экзаменуемый в целом правильно представляет путь решения задачи независимо от эффективности. При этом программа может быть представлена отдельными фрагментами, без ограничений на количество синтаксических и содержательных ошибок. 1 балл ставится также за решения, верные лишь в частных случаях	1
Не выполнены критерии, позволяющие поставить 1 или 2 балла	0
Максимальный балл для задания А	*2*
Критерии оценивания задания Б	
Программа правильно работает для любых соответствующих условию входных данных и при этом эффективна как по времени, так и по памяти, т.е. не используются массивы и другие структуры данных, размер которых зависит от количества входных элементов, а время работы пропорционально этому количеству. Возможно использование массивов и динамических структур данных (например, контейнеры STL в программе на языке C++) при условии, что в них в каждый момент времени хранится фиксированное количество элементов, требующих для хранения меньше 1кб (минимально необходимое количество – семь; допускается решение с запасом). Программа может содержать не более трёх синтаксических ошибок следующих видов: – пропущен или неверно указан знак пунктуации (запятая, точка с запятой, скобки и т.д.); – неверно написано или пропущено служебное слово языка программирования; – не описана или неверно описана переменная; – применяется операция, недопустимая для соответствующего типа данных. К синтаксическим ошибкам приравнивается использование неверного типа данных (например, использование целого типа вместо вещественного для представления данных при вводе и обработке). Если одна и та же ошибка встречается несколько раз, она считается за одну ошибку	4
Не выполнены условия, позволяющие поставить 4 балла. Программа правильно работает для любых соответствующих условию входных данных, время работы пропорционально количеству входных элементов. Размер используемой памяти не имеет значения и может зависеть от объёма входных данных. В частности, допускается использование одного или нескольких массивов размера N (как в приведённой выше программе 3).	3

Программа может содержать не более пяти синтаксических и приравненных к ним ошибок, описанных в критериях на 4 балла. Кроме того, допускается наличие не более одной содержательной ошибки из числа следующих: неверная инициализация при поиске минимального значения; неверная обработка начальных элементов данных, которая может, например, привести к получению ошибочного ответа при $7 < N < 14$; неточное определение границ массива, выход за границу массива (например, описан массив с границами от 1 до 7, а реально используется от 0 до 6 или наоборот); вычисленный индекс элемента массива на 1 отличается от верного; используется операция ”<” вместо ”<=”, ”or” вместо ”and” и т.п.; не учитывается, что заданные показания могут начинаться с одного или нескольких чётных чисел; не учитывается, что для данного набора показаний может не быть ни одного удовлетворяющего условиям произведения	
Не выполнены условия, позволяющие поставить 3 или 4 балла. Программа работает в целом верно, эффективно или нет. Например, допускается переборное решение, аналогичное приведённой выше программе 4. Допускается до семи синтаксических и приравненных к ним ошибок (см. критерии на 4 балла). Допускается до двух содержательных ошибок, описанных в критериях на 3 балла	2
Не выполнены условия, позволяющие поставить 2, 3 или 4 балла. Из описания алгоритма или общей структуры программы видно, что экзаменуемый в целом правильно представляет путь решения задачи независимо от эффективности. При этом программа может быть представлена отдельными фрагментами, без ограничений на количество синтаксических и содержательных ошибок. 1 балл ставится также за решения, верные лишь в частных случаях	1
Не выполнены критерии, позволяющие поставить 1, 2, 3 или 4 балла	0
Максимальный балл для задания Б	*4*
Итоговый максимальный балл	*4*