Ключи к районной репетиционной работе по информатике и ИКТ в форме единого государственного экзамена в 11-х классах

<table>
<tr><th></th><th>1 вариант</th><th></th><th>2 вариант</th></tr>
<tr><td>1.</td><td>2</td><td>1.</td><td>3</td></tr>
<tr><td>2.</td><td>wxyz</td><td>2.</td><td>yxwz</td></tr>
<tr><td>3.</td><td>40</td><td>3.</td><td>8</td></tr>
<tr><td>4.</td><td>5</td><td>4.</td><td>2</td></tr>
<tr><td>5.</td><td>14</td><td>5.</td><td>24</td></tr>
<tr><td>6.</td><td>115</td><td>6.</td><td>103</td></tr>
<tr><td>7.</td><td>203</td><td>7.</td><td>36</td></tr>
<tr><td>8.</td><td>32</td><td>8.</td><td>16</td></tr>
<tr><td>9.</td><td>16</td><td>9.</td><td>256</td></tr>
<tr><td>10.</td><td>КУААК</td><td>10.</td><td>324</td></tr>
<tr><td>11.</td><td>17</td><td>11.</td><td>34</td></tr>
<tr><td>12.</td><td>192</td><td>12.</td><td>20</td></tr>
<tr><td>13.</td><td>275</td><td>13.</td><td>170</td></tr>
<tr><td>14.</td><td>1</td><td>14.</td><td>572</td></tr>
<tr><td>15.</td><td>16</td><td>15.</td><td>16</td></tr>
<tr><td>16.</td><td>7</td><td>16.</td><td>34</td></tr>
<tr><td>17.</td><td>54</td><td>17.</td><td>910</td></tr>
<tr><td>18.</td><td>99</td><td>18.</td><td>18</td></tr>
<tr><td>19.</td><td>5</td><td>19.</td><td>5</td></tr>
<tr><td>20.</td><td>107</td><td>20.</td><td>35</td></tr>
<tr><td>21.</td><td>235</td><td>21.</td><td>82</td></tr>
<tr><td>22.</td><td>42</td><td>22.</td><td>50</td></tr>
<tr><td>23.</td><td>162</td><td>23.</td><td>162</td></tr>
<tr><td>24.</td><td>1. При вводе числа 1984 программа выведет 8.
Комментарий для экспертов. Приведённая программа вместо подсчёта суммы чётных цифр запоминает очередную чётную цифру, забывая при этом предыдущие. Поскольку цифры в записи числа обрабатываются с конца (справа налево), программа запоминает и выводит первую чётную цифру в десятичной записи или 0, если чётных цифр нет.
2. Примеры чисел, для которых программа даёт верный ответ: 1975, 1961, 30051,2013.
Комментарий для экспертов. Программа выдает верный ответ в следующих случаях:
1. В числе вообще нет чётных цифр.
2. В числе ровно одна чётная цифра.
3. Все чётные цифры числа-нули.
4. В числе есть нули и ровно одна ненулевая чётная цифра, причём все нули расположены правее ненулевой четной цифры.
3. Ошибка содержится ровно в одной строке программы: вместо присваивания переменной s значения найденной чётной цифры нужно увеличить s на значение этой цифры.
<table>
<tr><th>Бейсик</th><th>Паскаль</th><th>Python</th></tr>
<tr><td>Строка с ошибкой:
S = N MOD 10

Возможные варианты исправления:
S = S + N MOD 10
S = N MOD 10 + S</td><td>Строка с ошибкой:
S := N mod 10;

Возможные варианты исправления:
s := s + N mod 10;
s := N mod 10 + s;
Точка с запятой в конце строки не обязательна.</td><td>Строка с ошибкой:
s = N % 10

Возможные варианты исправления:
s += N % 10
s = s + N % 10
s = N % 10 + s</td></tr>
<tr><td>Си</td><td>Алгоритмический</td><td></td></tr>
<tr><td>Строка с ошибкой:
S = N % 10;</td><td>Строка с ошибкой:
s := mod(N,10)</td><td></td></tr>
</table>
</td><td>24.</td><td>Решение использует запись программы на Паскале. Допускается использование программы на трёх других языках.
1. Программа выведет число 9.
2. Первая ошибка. Неверная инициализация ответа (переменная max_digit).
Строка с ошибкой:
max_digit := 9;
Возможные варианты исправления:
max_digit := 0;
Возможны и другие исправления инициализации, например, на отрицательное число, в том числе -maxint.
3. Вторая ошибка. Неверное условие продолжения цикла. Программа не будет рассматривать старшую цифру числа.
Строка с ошибкой:
while N >= 10 do
Возможные варианты исправления:
while (N >= 1) do
или
while (N > 0) do
При этом замены на
while (N > 1) do или while (N >= 0) do
корректными не являются</td></tr>
</table>

Возможные варианты исправления:
s = s + N % 10;
s = N % 10 + s;
s += N % 10;
	Возможные варианты исправления:
	s := s + mod(N,10)
	s := mod(N,10) + s	

В любом варианте допустимы избыточные скобки, не изменяющие правильный порядок действий.

Некоторые строки программы могут показаться ошибочными, но в действительности ошибок не содержат. В первую очередь, к ним относятся строки с условиями.

Условие цикла (N > 1) может показаться неправильным. Действительно в стандартной схеме решения подобных задач используется условие N>0. Условие N > 1 приводит к тому, что, если старшая цифра числа равна 1, она не будет обрабатываться. Однако, поскольку 1 — нечётная цифра, ее обработка никак не влияет на результат, поэтому в данном случае такое условие допустимо.

Проверка чётности (N mod 2 = 0) может показаться неправильной, так как проверяется чётность не последней цифры, а числа в целом. Для проверки последней цифры следовало бы написать N mod 10 mod 2 = 0. Однако известно, что чётность числа всегда совпадает с чётностью его последней цифры, поэтому приведённая в программе проверка не может считаться ошибкой.

25.

ПАСКАЛЬ	БЕЙСИК

max:=a [1] ;
max2 : = a [2] ;
if max < max2 then
begin
max : = a [2] ;
max2 := a [1]
end;
for i : = 3 to N do
if a[i] > max then
begin
max2 : = max;
max := a [i]
end
else
if a[i] > max2 then
max2 := a[i];
writeln(max2);
	MAX = A(1)
	MAX2 = A(2)
	IF MAX < MAX2 THEN
	MAX = A(2)
	MAX2 = A(1)
	ENDIF
	FOR I = 3 TO N
	IF A(I) > MAX THEN
	MAX2 = MAX
	MAX = A (I)
	ELSE
	IF A(I) > MAX2 THEN
	MAX2 = A(I)
	ENDIF
	ENDIF ,
	NEXT I
	PRINT MAX2

СИ	Алгоритмический язык

max = a [0] ;
max2 = a [1] ;
if (max < max2)
{
max = a [1] ;
max2 = a [0] ;
}
for(i = 2; i < N; i++)
if(a[i] > max)
{
max 2 = max;
max = a [i];
}
else
if (a [i] > max2)
max2 = a [i] ;
cout « max2 « endl;
	MAX := a [1]
	MAX2 := a [2]
	если MAX < MAX2
	то
	MAX := a [2]
	MAX2 := a [1]
	все
	нц для i от 3 до N
	если a[i]>MAX
	то
	MAX2 :"= MAX
	MAX := a [i]
	иначе
	если a[i]>MAX2
	то
	MAX2 := a [i]
	все
	все
	кц
	вывод MAX2

25.

На языке Паскаль

```
k := 0;
for i := 1 to N - 1 do
   if ((a[i] + a[i + 1] ) mod 2 = 0) and
(a[i] * a[i + 1] > 100)
   then
      inc(k);
writeln(k);
```

На алгоритмическом языке

```
k := 0
нц для i от 1 до N − 1
   если mod(a[i] + a[i + 1], 2) = 0 и a[i]
* a[i + 1] > 100
   то
      k := k + 1
   все
кц
вывод k
```

На языке Бейсик

```
K = 0
FOR I = 1 TO N−1
   IF (A(I) + A(I + 1)) MOD 2 = 0 AND
A(I) * A(I + 1) > 100 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] + a[i + 1]) % 2 == 0 && a[i]
* a[i + 1] > 100)
      k++;
cout << k;
```

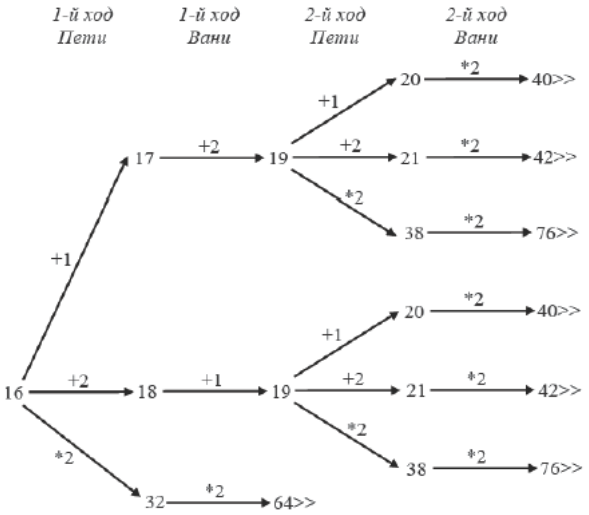
На языке Python

```
k = 0
for i in range(N - 1):
   if (a[i] + a[i + 1]) % 2 == 0 and a[i] *
a[i + 1] > 100:
      k += 1
print(k)
```

26.

1. а) Петя может выиграть, если S = 20, ... 38. Во всех этих случаях достаточно удвоить количество камней. При меньших значениях S за один ход нельзя получить кучу, в которой больше 38 камней.
б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче будет S = 19 камней. Тогда после первого хода Пети в куче будет 20 камней или 21 камень. Во всех случаях Ваня удваивает количество камней и выигрывает первым ходом.
2. Возможные значения S: 17, 18. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 19 камней: в первом случае добавлением двух камней, во втором добавлением одного камня. Эта позиция разобрана в п. 16. В ней игрок, который будет ходить (теперь это Ваня), выиграть не может, а его противник (то есть Петя) следующим ходом выиграет.
3. Возможное значение S: 16. После первого хода Пети в куче будет 17, 18 или 32 камня. Если в куче станет 32 камня, Ваня удвоит количество камней и выиграет первым ходом. Ситуация, когда в куче 17 или 18 камней, уже разобрана в п. 2. В этих ситуациях игрок, который будет ходить (теперь это Ваня), выигрывает своим вторым ходом.
В таблице изображено дерево возможных партий при описанной стратегии Вани. Заключительные позиции (в них выигрывает Ваня) подчёркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

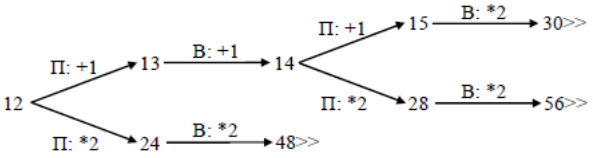
	Положения после очередных ходов			
И.п.	1-й ход Пети (разобраны все ходы)	1-й ход Вани (только ход по стратегии)	2-й ход Пети (разобраны все ходы)	2-й ход Вани (только ход по стратегии)
16	16+1=**17**	17+2=**19**	19+1=**20**	<u>20*2=**40**</u>
			19+2=**21**	<u>21*2=**42**</u>
			19*2=**38**	<u>38*2=**76**</u>
	16+2=**18**	18+1=**19**	19+1=**20**	<u>20*2=**40**</u>
			19+2=**21**	<u>21*2=**42**</u>
			19*2=**38**	<u>38*2=**76**</u>
	16*2=**32**	<u>24*2=**64**</u>		



26.

1. а) Петя может выиграть, если S = 15, … 28. Во всех этих случаях достаточно удвоить количество камней. При меньших значениях S за один ход нельзя получить кучу, в которой больше 28 камней.
б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче будет S = 14 камней. Тогда после первого хода Пети в куче будет 15 или 28 камней. Во втором случае Ваня удваивает количество камней и выигрывает в один ход.
2. Возможные значения S: 7 и 13. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 14 камней: в первом случае удвоением, во втором добавлением одного камня. Эта позиция разобрана в п. 1б. В ней игрок, который будет ходить (теперь это Ваня), выиграть не может, а его противник (то есть Петя) следующим ходом выиграет.
3. Возможное значение S: 12. После первого хода Пети в куче будет 13 камней или 24 камня. Если в куче станет 24 камня, Ваня удвоит количество камней и выиграет первым ходом. Ситуация, когда в куче 13 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 при описанной стратегии Вани. Заключительные позиции (в них выигрывает Ваня) подчёркнуты. На рисунке это же дерево изображено в графическом виде (оба способа изображения дерева допустимы).

И.п.	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)
12	12 + 1 = **13**	13 + 1 = **14**	14 + 1 = **15**	15*2 = **30**
			14*2 = **28**	28*2 = **56**
	12*2 = **24**	24*2 = **48**		



27.

Произведение двух чисел делится на 26, если выполнено одно из следующих условий (условия не могут выполняться одновременно).
А. Оба сомножителя делятся на 26.
Б. Один из сомножителей делится на 26, а другой не делится.
В. Ни один из сомножителей не делится на 26, но один сомножитель делится на 2, а другой – на 13.
Примечание для проверяющего. Условие делимости произведения на 26 можно сформулировать проще, например, так: (один из сомножителей делится на 26) ИЛИ (один сомножитель делится на 2, а другой – на 13). Но в этом случае пара сомножителей может удовлетворять обоим условиям, что затруднит подсчёт количества пар.
При вводе чисел можно определять, делится ли каждое из них на 26, 2 и 13, и подсчитывать следующие значения:
1) n26 – количество чисел, кратных 26;
2) n13 – количество чисел, кратных 13, но не кратных 26;

27.

Заметим, что сумма двух элементов кратна 120 тогда, когда сумма их остатков от деления на 120 будет равна 120 (или равна 0, если оба числа кратны 120). Необходимо хранить в массиве максимальные числа с остатками от 0 до 119, а каждое новое введённое число складывать с числами из массива и искать наибольшую сумму, кратную 120.
Пример правильной и эффективной программы на языке Паскаль:

```
var
  a: array[0..119] of integer;
  i, j, n, maxs, x1, x2, n1, n2: integer;
begin
  readln(n);
  for i := 0 to 119 do
    a[i] := 0;
  maxs := 0;
  readln(x1);
```

3) n2 – количество чисел, кратных 2, но не кратных 26.

Примечание для проверяющего. Сами числа при этом можно не хранить. Каждое число учитывается не более чем в одном из счётчиков. Количество пар, удовлетворяющих условию А, можно вычислить по формуле n26·(n26 – 1)/2.

Количество пар, удовлетворяющих условию Б, можно вычислить по формуле n26·(N – n26).

Количество пар, удовлетворяющих условию В, можно вычислить по формуле n2·n13.

Поэтому искомое количество пар вычисляется по формуле n26·(n26 – 1)/2 + n26·(N – n26) + n2·n13.

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

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

```
var
    N: integer; {количество чисел}
    a: integer; {очередное число}
    n26, n13, n2: integer;
    k26: integer; {количество требуемых пар}
    i: integer;

begin
    readln(N);
    n26:=0; n13:=0; n2:=0;
    for i:=1 to N do begin
       readln(a);
       if a mod 26 = 0 then
          n26 := n26+1
      else if a mod 13 = 0 then
          n13 := n13 + 1
       else if a mod 2 = 0 then
          n2 := n2 + 1;
   end;
   k26 := n26*(n26-1) div 2 + n26*(N-n26) + n2*n13;
   writeln(k26)
end.
```

Пример 2. Программа на языке Паскаль. Программа не эффективна по времени и по памяти

```
var N: integer;
   a: array[1..1000] of integer;
   i, j, k: integer;
begin
   readln(N);
   for i:=1 to N do read(a[i]);
   k:= 0;
   for i:= 1 to N-1 do
      for j:= i+1 to N do
         if a[i]*a[j] mod 26 = 0 then
            k := k + 1;
      writeln(k)
end.
```

```
   for i := 2 to n do begin
      readln(x2);
      if (x1 > a[x1 mod 120]) then
         a[x1 mod 120] := x1;
      for j := 0 to 119 do
         if (a[j] + x2 > maxs) and ((a[j] + x2) mod 120 = 0) and
(a[j] > x2) then begin
            maxs := a[j] + x2;
            n1 := a[j];
            n2 := x2;
         end;
      x1 := x2;
   end;
   if maxs > 0 then
      writeln(n1, ' ', n2)
   else
      whiteln("NO");
end.
```