Вариант 1

24. 1. Например, $x = 0$, $y = 2$. Подойдёт любая точка, у которой $y > 0$ или $y < -1$ или ($y <= 0$ и $y >= -1$ и $y <= \cos(x)$ и $|x| >= 3.14$).

2. Возможная доработка (Паскаль):

```
if (y<=0) and (y>=-1) and (y<=cos(x)) and (x>=-3.14) and (x<=3.14)
then write('принадлежит')
else write('не принадлежит')
```

Возможны и другие способы решения.

25. *Решение на естественном языке*:

Объявим константу N, равную числу элементов массива (30), целочисленные переменные s для хранения текущей суммы отрицательных элементов, m для подсчёта количества отрицательных переменных и i для хранения индекса просматриваемого элемента. Ввод значений в массив описывать не требуется, так как он задан. Присвоим s и m значение ноль. В цикле от первого (нулевого) до последнего (N или $N - 1$) значения индекса совершим следующие действия.

Сравним с нулём значение текущего элемента массива. В случае, если значение текущего элемента меньше нуля, прибавим к текущему значению переменной s значение рассматриваемого элемента массива и увеличим значение переменной m на единицу.

По завершении цикла сравним значение переменной m с нулём (проверка наличия отрицательных элементов) и в случае, если значение m больше нуля, выводим значение переменной s, в ином случае выводим «Отрицательных элементов нет».

Замечание: можно обойтись без переменной m. В этом случае сообщение «Отрицательных элементов нет» выводится в случае, если значение s равно нулю.

Примеры записи алгоритма на языках программирования:

Бейсик

```
N=30
DIM A(N), S, M, I AS
INTEGER
S=0
M=0
FOR I=1 TO N
IF A(I) < 0 THEN
S = S + A(I)
M = M + 1
ENDIF
NEXT I
IF M > 0 THEN
PRINT S
ELSE
PRINT "Отрицательных
элементов нет"
ENDIF
END
```

Паскаль

```
const N=30;
var a: array [1..N] of
integer;
s, m, i: integer;
begin
s := 0;
m := 0;
for i:= 1 to N do
 if a[i] < 0 then
 begin
 s := s + a[i];
 m := m + 1;
 end;
if m >0
 then write (s)
 else write('Отрицательных
элементов нет');
end.
```

Си

```
void main(void)
{ const N=30;
int array [N];
int s, m;
s = 0;
m = 0;
for (int i=0;i<N;i++)
{
if (array [i]<0)
 { s+=array[i];
 m++;
 }
}
if (m>0) printf ("%d", s);
else printf
("Отрицательных элементов
нет");
}
```

26. 1. а) Петя может выиграть, если $S = 16, \ldots 47$. Во всех этих случаях достаточно утроить количество камней. При меньших значениях S за один ход нельзя получить кучу, в которой больше 47 камней.

б) Ваня может выиграть первым ходом (как бы ни играл Петя), если исходно в куче будет $S = 15$ камней. Тогда после первого хода Пети в куче будет 16 или 45 камней.

В обоих случаях Ваня утраивает количество камней и выигрывает в один ход.

2. Возможные значения S: 5 и 14. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 15 камней: в первом случае утроением, 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: 13. После первого хода Пети в куче будет 14 или 39 камней. Если в куче станет 39 камней, Ваня утроит количество камней и выиграет первым ходом. Ситуация, когда в куче 14 камней, уже 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) выделены жирным шрифтом.

	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)**
13	**13+1=14**	**14+1=15**	**15+1=16**	**16*3=48**
			15*3=45	**45*3=135**
	13*3=39	**39*3=117**		

27. Программа верно читает входные данные, не запоминая их все, а сразу подсчитывая в массиве, хранящем 99 целых чисел согласно номерам школ, количество участников олимпиады из каждой школы. Затем ищется наибольший элемент в данном массиве, затем распечатываются номера соответствующих школ, одновременно подсчитывая их количество.

```
var nc:array[1..99] of integer;
 p:1..99;
 c:char;
 i, k, N, max: integer;
begin
 readln(N);
 for i:=0 to 99 do nc[i]:=0;
 for i:=1 to N do
 begin
  repeat
   read(c)
  until c=' '; {считана фамилия}
  repeat
   read(c)
  until c=' '; {считаны инициалы}
  readln(p);
  nc[p]:=nc[p]+1;
 end;
 max:=0;
 for i:=1 to 99 do
  if nc[i]>max then max:=nc[i];
 k:=0;
 for i:=1 to 99 do
 if nc[i]=max then
 begin
  writeln(i);
  k:=k+1
```

```
 end;
 writeln('Количество школ, приславших наибольшее число участников', k)
end.
```

Вариант 2

24. 1. Например, $x = 1$, $y = -1$. Подойдёт любая точка, у которой $y < 0$ или $x < 0$ или ($y >= 0$ и $y <= \cos(x)$ и $x > 1.575$).

2. Возможная доработка (Паскаль):

```
if (y>=0) and (x<=1.575) and (y<=cos(x)) and (x>=0)
then write('принадлежит')
else write('не принадлежит')
```

Возможны и другие способы решения.

25. *Решение на естественном языке*:

Объявим константу N, равную числу элементов массива (30), целочисленные переменные p для хранения текущего произведения отрицательных элементов и i для хранения индекса просматриваемого элемента. Ввод значений в массив описывать не требуется, так как он задан. Присвоим p значение единица. В цикле от первого (нулевого) до последнего (N или $N - 1$) значения индекса совершим следующие действия.

Сравним с нулём значение текущего элемента массива. В случае, если значение текущего элемента меньше нуля, умножим текущее значение переменной p на значение рассматриваемого элемента массива.

По завершении цикла выводим значение переменной p.

Примеры записи алгоритма на языках программирования:

Бейсик	Паскаль	Си

N=30
DIM A(N), P, I AS
INTEGER
P = 1
FOR I=1 TO N
IF A(I) < 0 THEN
P = P * A(I)
ENDIF
NEXT I
PRINT P
END
	const N=30;
	var a: array [1..N] of
	integer;
	p, i: integer;
	begin
	p := 1;
	for i:= 1 to N do
	if a[i] < 0 then
	p := p * a[i];
	writeln (p);
	end.
		void main(void)
		{ const N=30;
		int array [N];
		int p;
		p = 1;
		for (int i=0; i<N; i++)
		{ if (array [i]<0)
		{ p= p * array[i];
		}
		}
		printf ("%d", p);
		}

26. 1. а) Петя может выиграть, если $S = 13, \ldots 38$. Во всех этих случаях достаточно утроить 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 $S = 12$ камней. Тогда после первого хода Пети в куче будет 13 или 36 камней. В обоих случаях Ваня утраивает количество камней и выигрывает в один ход.

2. Возможные значения S: 4 и 11. В этих случаях Петя, очевидно, не может выиграть первым ходом. Однако он может получить кучу из 12 камней: в первом случае утроением, 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: 10. После первого хода Пети в куче будет 11 или 30 камней. Если в куче станет 30 камней, Ваня утроит количество камней и выиграет первым ходом. Ситуация, когда в куче 11 камней, уже 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) выделены жирным шрифтом.

	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)**
10	**10+1=11**	**11+1=12**	**12+1=13**	**13*3=39**
			12*3=36	**36*3=108**
	10*3=30	**30*3=90**		

27. Программа верно читает входные данные, не запоминая их все, а сразу подсчитывая в массиве, хранящем 99 целых чисел согласно номерам школ, количество участников олимпиады из каждой школы. Затем подсчитывается количество школ, приславших хотя бы одного участника, и вычисляется среднее количество участников от одной школы.

```
var nc:array[1..99] of integer;
 p:1..99;
 c:char;
 i, k, N: integer;
begin
 readln(N);
 for i:=1 to 99 do nc[i]:=0;
 for i:=1 to N do
 begin
  repeat
   read(c)
  until c=' '; {считана фамилия}
  repeat
   read(c)
  until c=' '; {считаны инициалы}
  readln(p);
  nc[p]:=nc[p]+1;
 end;
 k:=0;
 for i:=1 to 99 do
  if nc[i]>0 then k:=k+1;
 writeln('Среднее количество участников из одной школы', N/k)
end.
```

Вариант 5

24. 1) $a = -1$, $b = 0$, $x = 0$.

Значение x может быть не указано. Значение a может быть любым отрицательным числом. Также допустим ответ, что программа работает неправильно при любых отрицательных (a и $b = 0$).

2) Лишняя часть:
не нужно вводить x с клавиатуры;
верно: readln(a, b).
3) Возможная доработка:

```
readln(a, b);
if b = 0 then
if a > 0 then
write('x > 0 или x < 0')
else
write('нет решений')
else
if a > 0 then
write('x > 0 или x <', -b)
else
write(-b, '< x < 0');
```

(могут быть и другие способы доработки).

25. **Введём целочисленную переменную SumNeg и целочисленную переменную NumNeg, в которые будем заносить соответственно сумму и число отрицательных элементов в просмотренной части массива, и присвоим им значение 0. В цикле до конца массива: проверяем, является ли очередной элемент отрицательным. Если да, то прибавляем его к SumNeg и увеличиваем счётчик NumNeg на единицу. По окончании цикла выводим SumNeg/NumNeg.**

Пример правильной и эффективной программы **(на основе алгоритма, использующего однократный проход по массиву):**

На языке Паскаль	На языке Бейсик

Const N = 30;
Var a:array [1..N] of integer;
SumNeg, NumNeg, I: integer;
begin
SumNeg :=0;
NumNeg :=0;
for I := 1 to N do
if a[I]<0 then
begin
SumNeg := SumNeg + a[I];
NumNeg := NumNeg + 1;
end;
writeln (SumNeg/NumNeg);
end.
	N = 30
	DIM I, SumNeg, NumNeg, A(N) AS INTEGER
	SumNeg=0
	NumNeg=0
	FOR I = 1 TO N
	IF A(I)<0 THEN
	SumNeg = SumNeg + A(I)
	NumNeg = NumNeg + 1
	ENDIF
	NEXT I
	PRINT SumNeg/NumNeg
	END

26. **Выигрывает второй игрок.**

Для доказательства рассмотрим неполное дерево игры, оформленное в виде таблицы, где в каждой ячейке записаны пары чисел, разделённые запятой. Эти числа соответствуют количеству камней на каждом этапе игры, в первой и второй кучах соответственно.

	1-й ход	2-й ход	3-й ход	4-й ход	
Стартовая позиция	**I-й игрок (все варианты хода)**	**II-й игрок (выигрышный ход)**	**I-й игрок (все варианты хода)**	**II-й игрок (один из вариантов)**	**Пояснение**
6,5	12,5	12,10	24,10	72,10	Второй игрок выигрывает на четвёртом ходу, после любого ответа первого игрока, например, утроив число камней в самой большой куче
			36,10	108,10	
			12,20	12,60	
			12,30	12,90	
	6,10	12,10	Те же варианты третьего-четвёртого ходов		
	18,5	54,5	Второй игрок выигрывает ответным ходом		
	6,15	6,45	Второй игрок выигрывает ответным ходом		

Таблица содержит *все возможные* варианты ходов первого игрока. Из неё видно, что при любом ходе первого игрока у второго имеется ход, приводящий к победе.

27. Программа считывает входные данные, сразу подсчитывая в массиве, хранящем 12 вещественных чисел, сумму температур в каждом из месяцев. Затем с использованием этого массива ищется максимальная среднемесячная температура. За дополнительный просмотр среднемесячных температур (их можно как запомнить в массиве, так и вычислить заново) распечатывается информация об искомых месяцах. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, месяц с максимальной температурой единственен).

Пример правильной и эффективной программы на языке Паскаль:

```
const d:array[1..12] of integer =
 (31,28,31,30,31,30,31,31,30,31,30,31);
var m:array[1..12] of real;
 max,t:real;
 i,j:integer;
 c1,c2:char;
begin
 for j:=1 to 12 do
  m[j]:=0;
 for i:=1 to 365 do
 begin
  readln(c1,c1,c1,c1,c2,t);
  j:=(ord(c1)-ord('0'))*10+
   ord(c2)-ord('0');
 m[j]:=m[j]+t
 end;
 max:=m[1]/d[1];
 for j:=2 to 12 do
  if m[j]/d[j] > max then
   max:=m[j]/d[j];
 for j:=1 to 12 do
  if abs(m[j]/d[j]-max) < 0.0001
  then writeln(j,' ',m[j]/d[j]:0:1)
end.
```

Пример правильной программы на языке Бейсик:

```
DATA 31,28,31,30,31,30,31,31,30,31,30,31
DIM i, j, d(12) AS INTEGER
DIM m(12)
DIM dat AS STRING * 5
FOR i = 1 TO 12
 m(i) = 0
 READ d(i)
NEXT i
FOR i = 1 TO 365
 INPUT dat, t
 j = (ASC(MID$(dat, 4, 1)) - ASC("0")) * 10 + ASC(MID$(dat, 5, 1)) - ASC("0")
 m(j) = m(j) + t
NEXT i
max = m(1) / d(1)
FOR j = 2 TO 12
 IF m(j) / d(j) > max THEN max = m(j) / d(j)
NEXT j
FOR j = 1 TO 12
 IF ABS(m(j) / d(j) - max) < .0001 THEN
  PRINT j; " ";
  PRINT USING "##.#"; m(j) / d(j)
 ENDIF
NEXT j
END
```

Вариант 6

24. **1) $a = 1$, $b = -1$, $x = 0$.**

Значение x может быть не указано. Значения a и b могут быть любыми ненулевыми числами с разными знаками. Ошибка программиста состоит в том, что программа работает неправильно при любых ненулевых a и b, имеющих разные знаки.

2) Лишняя часть:
не нужно вводить x с клавиатуры;
верно: readln(a, b).

3) Возможная доработка:

```
readln(a, b);
if a = 0 then
if b = 0 then write('любое число')
else write('нет решений')
else
if b/a > 0 then
write('x =', b/a, ' или x =', -b/a)
else
if b = 0 then write('x = 0')
else write('нет решений');
```

(могут быть и другие способы доработки).

25. *Пример правильного описания алгоритма на русском языке.*

Заводим переменную MaxNeg для хранения максимального количества подряд идущих отрицательных элементов и счётчик NumNeg для хранения числа отрицательных элементов в последней группе отрицательных элементов. Просматривая элементы массива, сравниваем очередной элемент с 0. Если очередной элемент массива оказывается неот-

рицательным, то сравниваем текущее значение счётчика NumNeg со значением переменной MaxNeg; если он больше, то заменяем значение переменной MaxNeg значением счётчика, при этом значение NumNeg обнуляется. Так повторяем до конца массива. В конце работы нужно еще раз сравнить значение счётчика со значением переменной MaxNeg и переопределить её, если счётчик больше.

Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль	На языке Бейсик

const N=30;
var a:array[1..N] of integer;
MaxNeg, NumNeg, i: integer;
begin
  MaxNeg:=0;
  NumNeg:=0;
  for i:=1 to N do
  begin
  if a[i]<0 then NumNeg:=NumNeg+1
  else begin
  if NumNeg> MaxNeg then
  MaxNeg:=NumNeg;
  NumNeg:=0;
  end;
  end;
  if NumNeg> MaxNeg then
  MaxNeg:=NumNeg;
  writeln(MaxNeg);
end.
	N=30
	DIM i, MaxNeg, NumNeg, a(N) AS INTEGER
	MaxNeg=0
	NumNeg=0
	FOR i = 1 TO N
	IF a(i)<0 THEN
	  NumNeg=NumNeg+1
	  ELSE
	  IF NumNeg>MaxNeg THEN
	  MaxNeg=NumNeg
	  ENDIF
	  NumNeg=0
	ENDIF
	NEXT i
	IF NumNeg>MaxNeg THEN
	  MaxNeg=NumNeg
	ENDIF
	PRINT MaxNeg
	END

26. Выигрывает второй игрок.

Для доказательства рассмотрим неполное дерево игры, оформленное в виде таблицы, где в каждой ячейке записаны пары чисел, разделённые запятой. Эти числа соответствуют количеству камней на каждом этапе игры в первой и второй кучах соответственно.

	1-й ход	2-й ход	3-й ход	4-й ход	
Стартовая позиция	I-й игрок (все варианты хода)	II-й игрок (выигрышный ход)	I-й игрок (все варианты хода)	II-й игрок (один из вариантов)	Пояснение
2,3	4,3	4,6	8,6	24,6	Второй игрок выигрывает на четвёртом ходу, после любого ответа первого игрока, например, утроив число камней в самой большой куче
			12,6	36,6	
			4,12	4,36	
			4,18	4,54	
	6,3	6,6	12,6	36,6	Второй игрок выигрывает на четвёртом ходу, после любого ответа первого игрока, например, утроив число камней в самой большой куче
			18,6	54,6	
			6,12	6,36	
			6,18	6,54	
	2,6	6,6	Те же варианты третьего-четвёртого ходов		
	2,9	2,27	Второй игрок выигрывает ответным ходом		

Таблица содержит *все возможные* варианты ходов первого игрока. Из неё видно, что при любом ходе первого игрока у второго имеется ход, приводящий к победе.

27. Программа читает входные данные, сразу подсчитывая минимальную длину встречающихся слов. За второй проход исходных данных производится замена букв латинского алфавита и печать расшифрованного сообщения.

Пример правильной и эффективной программы на языке Паскаль:

```
var f:boolean;
 i, k, min: integer;
 c,cnew:char;
 s:string;
begin
 s:='';
 min:=250; k:=0;
 f:=false;
 repeat
  read(c);
  s:=s+c;
  if f then {слово началось}
   if c in ['a'..'z','A'..'Z']
    then k:=k+1
    else begin
     if k<min then min:=k;
      f:=false
    end
 else {f=false}
  if c in ['a'..'z','A'..'Z']
   then begin f:=true; k:=1 end
 until c='.';
 for i:=1 to length(s) do
 begin
  cnew:=chr(ord(s[i])+min);
  case s[i] of
  'a'..'z':if cnew>'z' then write(chr(ord(cnew)-26))
    else write(cnew);
 'A'..'Z':if cnew>'Z' then write(chr(ord(cnew)-26))
    else write(cnew);
   else write(s[i])
  end;
 end;
 readln
end.
```

Пример правильной программы на языке Бейсик:

```
DIM i, j, min, k, f, a(26) AS INTEGER
DIM s AS STRING
INPUT s
i = 1
k = 0
min = 250
```

```
f = 0
WHILE NOT (MID$(s, i, 1) = ".")
 c$ = MID$(s, i, 1)
 IF f = 1 THEN
  IF (c$ >= "A") AND (c$ <= "Z") OR
   (c$ >= "a") AND (c$ <= "z") THEN
  k = k + 1
 ELSE IF k < min THEN min = k
   f = 0
  ENDIF
 ELSE
  IF (c$ >= "A") AND (c$ <= "Z") OR
   (c$ >= "a") AND (c$ <= "z") THEN
  f = 1: k = 1
  ENDIF
 ENDIF
 i = i + 1
WEND
IF k < min THEN min = k
FOR j = 1 TO i
 cnew$ = CHR$(ASC(MID$(s, j, 1)) + min)
 IF (MID$(s, j, 1) >= "a") AND (MID$(s, j, 1) <= "z") THEN
  IF cnew$ > "z" THEN
  PRINT (CHR$(ASC(cnew$) - 26));
  ELSE PRINT cnew$;
  ENDIF
 ELSE
 IF (MID$(s, j, 1) >= "A") AND (MID$(s, j, 1) <= "Z") THEN
  IF cnew$ > "Z" THEN
  PRINT (CHR$(ASC(cnew$) - 26));
  ELSE PRINT cnew$;
  ENDIF
 ELSE PRINT MID$(s, j, 1);
 ENDIF
 ENDIF
NEXT j
END
```

Вариант 7

24. **1) Пример: $x = -2$, $y = 1$.**
Любая пара (x, y), для которой выполняется: $x > 1.5$ или ($y >= 0$ и $x < 0$ и $y <= x^2$) или $y > x^2$.

2) Возможная доработка (Паскаль):

```
if (y <= x*x) and (x <= 1.5) and (y >= 0) and (x >= 0) then
write('принадлежит')
else
write('не принадлежит');
```

(могут быть и другие способы доработки).

25. *Пример правильного описания алгоритма на русском языке.*

Введём целочисленные переменные MaxEv и MaxOdd, в которые будем заносить соответственно значения максимального чётного и максимального нечётного элемента в просмотренной части массива, и присвоим им начальное значение 0. В цикле до конца массива: проверяем, является ли очередной элемент чётным. Если да, то сравниваем его с MaxEv, если он больше, заносим его значение в переменную MaxEv. Если же элемент нечётен, то сравниваем его с MaxOdd, если он больше, заносим его значение в переменную MaxOdd. По окончании цикла выводим разность MaxEv-MaxOdd.

Для определения чётности значения элемента массива можно воспользоваться либо стандартной функцией (`if not odd (a[I]) then…`), либо операцией определения остатка от деления на 2 (`if a[I] mod 2 = 0 then…`), либо, как приведено в программе ниже, операцией целочисленного деления.

Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль

```
Const N = 30;
Var a:array [1..N] of integer;
MaxEv, MaxOdd, I: integer;
begin
MaxEv :=0;
MaxOdd :=0;
for I := 1 to N do
begin
if (a[I] div 2)*2 = a[I] then
begin
if a[I] > MaxEv then
MaxEv := a[I];
end
else
if a[I] > MaxOdd then
MaxOdd := a[I];
end;
writeln (MaxEv- MaxOdd);
end.
```

На языке Бейсик

```
N = 30
DIM I, MaxEv, MaxOdd, A(N) AS INTEGER
MaxEv = 0
MaxOdd = 0
FOR I = 1 TO N
IF (A(I)\2)*2 = A(I) THEN
IF A(I) > MaxEv THEN
MaxEv = A(I)
ENDIF
ELSE
IF A(I) > MaxOdd THEN
MaxOdd = A(I)
ENDIF
ENDIF
NEXT I
PRINT MaxEv- MaxOdd
END
```

26. Выигрывает второй игрок.

Для доказательства рассмотрим неполное дерево игры, оформленное в виде таблицы, где в каждой ячейке записаны координаты фишки на каждом этапе игры.

	1-й ход	2-й ход	3-й ход	4-й ход
Стартовая позиция	I-й игрок (все варианты хода)	II-й игрок (выигрышный ход)	I-й игрок (все варианты хода)	II-й игрок (выигрышный ход, один из вариантов)
3,–5	3,0	6,0	9,0	12,0
			6,4	9,4
			6,5	9,5
	3,–1	3,3	6,3	9,3
			3,7	6,7
			3,8	6,8
	6,–5	9,–5	Второй игрок выигрывает ответным ходом	

Таблица содержит *все возможные* варианты ходов первого игрока. Из неё видно, что при любом ходе первого игрока у второго имеется ход, приводящий к победе.

27. Программа читает входные данные, сразу подсчитывая минимальную длину встречающихся слов. За второй проход исходных данных производится замена букв латинского алфавита и печать расшифрованного сообщения. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, для строчных английских букв и без циклического сдвига).

Пример правильной и эффективной программы на языке Паскаль:

```
var f:boolean;
 i, k, min: integer;
 c,cnew:char;
 s:string;
begin
 s:='';
 min:=250; k:=0;
 f:=false;
 repeat
 read(c);
 s:=s+c;
 if f then {слово началось}
 if c in ['a'..'z','A'..'Z']
 then k:=k+1
 else begin
 if k<min then min:=k;
 f:=false
 end
 else {f=false}
 if c in ['a'..'z','A'..'Z']
 then begin f:=true; k:=1 end
 until c='.';
 for i:=1 to length(s) do
 begin
 cnew:=chr(ord(s[i])-min);
 case s[i] of
 'a'..'z':if cnew<'a' then write(chr(ord(cnew)+26))
 else write(cnew);
 'A'..'Z':if cnew<'A' then write(chr(ord(cnew)+26))
 else write(cnew);
 else write(s[i])
 end;
 end;
 readln
end.
```

Пример правильной программы на языке Бейсик:

```
DIM i, j, min, k, f, a(26) AS INTEGER
DIM s AS STRING
INPUT s
i = 1
k = 0
min = 250
f = 0
WHILE NOT (MID$(s, i, 1) = ".")
 c$ = MID$(s, i, 1)
 IF f = 1 THEN
 IF (c$ >= "A") AND (c$ <= "Z") OR
 (c$ >= "a") AND (c$ <= "z") THEN
 k = k + 1
 ELSE IF k < min THEN min = k
 f = 0
 ENDIF
 ELSE
 IF (c$ >= "A") AND (c$ <= "Z") OR
 (c$ >= "a") AND (c$ <= "z") THEN
 f = 1: k = 1
 ENDIF
 ENDIF
 i = i + 1
WEND
IF k < min THEN min = k
FOR j = 1 TO i
 cnew$ = CHR$(ASC(MID$(s, j, 1)) - min)
 IF (MID$(s, j, 1) >= "a") AND (MID$(s, j, 1) <= "z") THEN
 IF cnew$ < "a" THEN
 PRINT (CHR$(ASC(cnew$) + 26));
 ELSE PRINT cnew$;
 ENDIF
 ELSE
 IF (MID$(s, j, 1) >= "A") AND (MID$(s, j, 1) <= "Z") THEN
 IF cnew$ < "A" THEN
 PRINT (CHR$(ASC(cnew$) + 26));
 ELSE PRINT cnew$;
 ENDIF
 ELSE PRINT MID$(s, j, 1);
 ENDIF
 ENDIF
NEXT j
END
```

Вариант 8

24. **1)** $a = -1$, $b = 1$, $x = 0$.

Значение x может быть не указано. Значение a может быть любым отрицательным числом, значение b — любым положительным. Также допустим ответ, что программа работает неправильно при любых положительных b и отрицательных a.

2) Лишняя часть:

не нужно вводить x с клавиатуры;

верно: readln(a, b).

3) Возможная доработка:

```
readln(a, b);
if b > 0 then
if a > 0 then
write('x > ', a,' или x < 0')
else
write('x < ', a,' или x > 0')
else
if a > 0 then
write('0 < x < ',a)
else
write(a,' < x < 0');
```

(могут быть и другие способы доработки).

25. ***Пример правильного описания алгоритма на русском языке.***

Заводим переменную MaxIncr для хранения максимального количества подряд идущих в порядке возрастания элементов и счётчик NumIncr для хранения числа элементов в последней группе возрастающих элементов. Просматривая элементы массива, сравниваем очередной элемент со следующим за ним. Если очередной элемент массива оказывается не меньше следующего, то сравниваем текущее значение счётчика со значением переменной MaxIncr; если он больше, то заменяем значение переменной MaxIncr значением счётчика, при этом значение NumIncr обнуляется. Так повторяем до конца массива. В конце работы нужно ещё раз сравнить значение счётчика со значением переменной MaxIncr и переопределить её, если счётчик больше.

Пример правильной и эффективной программы **(на основе алгоритма, использующего однократный проход по массиву):**

На языке Паскаль	**На языке Бейсик**

`const N = 30;`
`var a:array[1..N] of integer;`
`MaxIncr, NumIncr, i: integer;`
`begin`
`  MaxIncr:=0;`
`  NumIncr:=0;`
`  for i:=1 to N-1 do`
`  begin`
`   if a[i]<a[i+1] then`
`    NumIncr:=NumIncr+1;`
`   else begin`
`    if NumIncr> MaxIncr then`
`     MaxIncr:=NumIncr;`
`    NumIncr:=0;`
`   end;`
`  end;`
`  if NumIncr> MaxIncr then`
`   MaxIncr:=NumIncr;`
`  writeln(MaxIncr);`
`end.`
	`N = 30`
	`DIM i, MaxIncr, NumIncr, a(N) AS`
	`INTEGER`
	`MaxIncr=0`
	`NumIncr=0`
	`FOR i = 1 TO N-1`
	`IF a(i)<a(i+1) THEN`
	`  NumIncr=NumIncr+1`
	`  ELSE`
	`  IF NumIncr>MaxIncr THEN`
	`    MaxIncr=NumIncr`
	`  ENDIF`
	`  NumIncr=0`
	`ENDIF`
	`NEXT i`
	`IF NumIncr>MaxIncr THEN`
	`  MaxIncr=NumIncr`
	`ENDIF`
	`PRINT MaxIncr`
	`END`

26. Выигрывает второй игрок.

Для доказательства рассмотрим неполное дерево игры, оформленное в виде таблицы, где в каждой ячейке записаны пары чисел, разделённые запятой. Эти числа соответствуют количеству камней на каждом этапе игры в первой и второй кучах соответственно.

<table>
<tr><th></th><th>1-й ход</th><th>2-й ход</th><th>3-й ход</th><th>4-й ход</th><th></th></tr>
<tr><th>Стартовая позиция</th><th>I-й игрок (все варианты хода)</th><th>II-й игрок (выигрышный ход)</th><th>I-й игрок (все варианты хода)</th><th>II-й игрок (один из вариантов)</th><th>Пояснение</th></tr>
<tr><td rowspan="8">1,2</td><td rowspan="4">3,2</td><td rowspan="4">3,6</td><td>9,6</td><td>27,6</td><td rowspan="4">Второй игрок выигрывает на четвёртом ходу, после любого ответа первого игрока, например, утроив число камней в самой большой куче</td></tr>
<tr><td>7,6</td><td>21,6</td></tr>
<tr><td>3,18</td><td>3,54</td></tr>
<tr><td>3,10</td><td>3,30</td></tr>
<tr><td rowspan="4">5,2</td><td rowspan="4">5,6</td><td>15,6</td><td>45,6</td><td rowspan="4">Второй игрок выигрывает на четвёртом ходу после любого ответа первого игрока, например, утроив число камней в самой большой куче</td></tr>
<tr><td>9,6</td><td>27,6</td></tr>
<tr><td>5,18</td><td>5,54</td></tr>
<tr><td>5,10</td><td>5,30</td></tr>
<tr><td></td><td>1,6</td><td>3,6 или 5,6</td><td colspan="3">Те же варианты третьего-четвёртого ходов</td></tr>
</table>

Таблица содержит *все возможные* варианты ходов первого игрока. Из неё видно, что при любом ходе первого игрока у второго имеется ход, приводящий к победе.

27. Программа должна верно читать входные данные, не запоминая их все, а сразу подсчитывая в массиве, хранящем 12 целых чисел, количество учащихся в каждой из параллелей. Затем с использованием этого массива ищется параллель с максимальным числом учеников. За дополнительный просмотр этого массива распечатывается информация об искомых параллелях. Баллы начисляются только за программу, которая решает задачу хотя бы для частного случая (например, параллель с максимальным количеством учеников единственна).

Пример правильной и эффективной программы на языке Паскаль:

```
var pc:array[1..12] of integer;
 p:1..12;
 class:string[3];
 c:char;
 max, i, N:integer;
begin
 readln(N);
 for i:=1 to 12 do
  pc[i]:=0;
 for i:=1 to N do
 begin
  repeat
   read(c)
  until c=' '; {считана фамилия}
  repeat
   read(c)
  until c=' '; {считано имя}
  readln(class);
  {определяем номер параллели}
  if length(class)=2 then
 p:=ord(class[1])-ord('0') else
   p:=(ord(class[1])-ord('0'))*10+
    ord(class[2])-ord('0');
  pc[p]:=pc[p]+1;{учитываем ученика этой параллели}
 end;
 max:=0;
 for i:=1 to 12 do
  if pc[i]>max then max:=pc[i];
 writeln('Максимум учеников в параллели:',max);
 for i:=1 to 12 do
  if pc[i]=max then
   write(i,' ');
 readln
end.
```

Пример правильной программы на языке Бейсик:

```
DIM i, j, p, n, max, pc(12) AS INTEGER
DIM m(12)
DIM s AS STRING
FOR i = 1 TO 12
  pc(i) = 0
NEXT i
INPUT n
```

```
FOR j = 1 TO n
LINE INPUT s
c$ = MID$(s, 1, 1)
i = 1
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
i = i + 1
c$ = MID$(s, i, 1)
WHILE NOT (c$ = " ")
 i = i + 1
 c$ = MID$(s, i, 1)
WEND
s = MID$(s, i + 1, 3)
IF MID$(s, 2, 1) >= "0" AND MID$(s, 2, 1) <= "2" THEN
 p = (ASC(MID$(s, 1, 1)) - ASC("0")) * 10 +
 ASC(MID$(s, 2, 1)) - ASC("0")
ELSE
 p = ASC(MID$(s, 1, 1)) - ASC("0")
ENDIF
pc(p) = pc(p) + 1
NEXT j
max = 0
FOR i = 1 TO 12
 IF pc(i) > max THEN max = pc(i)
NEXT i
PRINT "Max = "; max
FOR i = 1 TO 12
 IF pc(i) = max THEN PRINT i; " ";
NEXT i
END
```

Вариант 9

24. **1) $a = 1$, $b = -1$, $x = 0$. Значение x может быть не указано. Значение b может быть любым отрицательным числом, значение a — любым положительным. Также допустим ответ, что программа работает неправильно при любых положительных a и отрицательных b.**

2) Лишняя часть:

не нужно вводить x с клавиатуры;

верно: readln (a, b).

3) Возможная доработка:

```
readln(a,b);
if a=0 then
if b>0 then
write('нет решений')
else
write('x>0 или x<0')
else
if b>0 then
write(-a,'<x<0')
else
```

```
write('x>0 или x<',-a);
```

(могут быть и другие способы доработки).

25. *Пример правильного описания алгоритма на русском языке.*

Заводим переменную MaxEven для хранения максимального количества подряд идущих чётных элементов и счётчик NumEven для хранения числа чётных элементов в последней группе чётных элементов. Просматривая элементы массива, проверяем последний элемент на чётность. Если очередной элемент массива оказывается нечётным, то сравниваем текущее значение счётчика NumEven со значением переменной MaxEven; если он больше, то заменяем значение переменной MaxEven значением счётчика, при этом значение NumEven обнуляется. Так повторяем до конца массива. В конце работы нужно ещё раз сравнить значение счётчика со значением переменной MaxEven и переопределить её, если счётчик больше.

Пример правильной и эффективной программы (на основе алгоритма, использующего однократный проход по массиву):

На языке Паскаль	На языке Бейсик

`const N=30;`
`var a:array[1..N] of integer;`
`MaxEven, NumEven, i: integer;`
`begin`
`  MaxEven:=0;`
`  NumEven:=0;`
`  for i:=1 to N do`
`  begin`
`  if a[i] mod 2 = 0 then`
`    NumEven:=NumEven+1`
`  else begin`
`    if NumEven> MaxEven then`
`        MaxEven:=NumEven;`
`      NumEven:=0;`
`    end;`
`  end;`
`  if NumEven> MaxEven then`
`    MaxEven:=NumEven;`
`  writeln(MaxNeg);`
`end.`
	`N=30`
	`DIM i, MaxEven, NumEven AS INTEGER`
	`DIM a(N) AS INTEGER`
	`MaxEven=0`
	`NumEven=0`
	`FOR i = 1 TO N`
	`IF a(i)<0 THEN`
	`  NumEven=NumEven+1`
	`ELSE`
	`  IF NumEven>MaxEven THEN`
	`    MaxEven=NumEven`
	`  ENDIF`
	`  NumEven=0`
	`ENDIF`
	`NEXT i`
	`IF NumEven>MaxEven THEN`
	`  MaxEven=NumEven`
	`ENDIF`
	`PRINT MaxEven`
	`END`

26. Выигрывает первый игрок, своим первым ходом он должен добавить 2 камня в первую кучу. Для доказательства рассмотрим неполное дерево игры, оформленное в виде таблицы, где в каждой ячейке записаны пары чисел, разделённые запятой. Эти числа соответствуют количеству камней на каждом этапе игры в первой и второй кучах соответственно.

<table>
<tr><th></th><th>2-й ход</th><th>3-й ход</th><th>4-й ход</th><th>5-й ход</th><th></th></tr>
<tr><th>Позиция после первого хода</th><th>II-й игрок (все варианты хода)</th><th>I-й игрок (выигрышный ход)</th><th>II-й игрок (все варианты хода)</th><th>I-й игрок (один из вариантов)</th><th>Пояснение</th></tr>
<tr><td rowspan="5">5,6</td><td rowspan="4">5,8</td><td rowspan="4">7,8</td><td>14,8</td><td>28,8</td><td rowspan="4">Первый игрок выигрывает на пятом ходу, после любого ответа второго игрока, например, удвоив число камней в самой большой куче</td></tr>
<tr><td>9,8</td><td>18,8</td></tr>
<tr><td>7,16</td><td>7,32</td></tr>
<tr><td>7,10</td><td>7,20</td></tr>
<tr><td>7,6</td><td>7,8</td><td colspan="3">Те же варианты четвёртого-пятого ходов</td></tr>
<tr><td>5,12</td><td>5,24</td><td colspan="3">Первый игрок выиграл</td></tr>
<tr><td>10,6</td><td>20,6</td><td colspan="3">Первый игрок выиграл</td></tr>
</table>

Таблица содержит *все возможные* варианты ходов второго игрока. Из неё видно, что при любом ответе второго игрока у первого имеется ход, приводящий к победе.

27. Программа читает все входные символы до точки один раз, подсчитывая в массиве, хранящем 26 целых чисел, количество каждой из букв. Сами входные символы при этом не запоминаются. В дополнительный массив, состоящий из 26 символов, заносятся буквы от «*a*» до «*z*». Затем элементы первого массива сортируются по неубыванию любым алгоритмом сортировки, параллельно переставляются и элементы второго массива (возможно использование одного массива записей, состоящих из двух полей). При этом элементы с равным числом вхождений символов местами не меняются. Во втором из отсортированных массивов пропускаются элементы, количество которых равно 0, остальные элементы печатаются подряд.

Баллы начисляются только за программу, которая решает задачу хотя бы для одного частного случая (например, для строк, состоящих не более чем из 255 символов).

Пример правильной и эффективной программы на языке Паскаль:

```
var a:array[0..25] of integer;
    m:array[0..25] of 'a'..'z';
```

```
    c: char;
    i, j, k: integer;
begin
   for i:=0 to 25 do
 begin
    a[i]:=0;
    m[i]:=chr(ord('a')+i)
 end;
 read(c);
 while c<>'.' do
 begin
  a[ord(c)-ord('a')] := a[ord(c)-ord('a')] + 1;
  read(c);
 end;
 for i:=1 to 25 do
 for j := 0 to 24 do
  if a[j] > a[j+1] then
  begin
   k:=a[j]; c:=m[j];
   a[j]:=a[j+1]; m[j]:=m[j+1];
   a[j+1]:=k; m[j+1]:=c
  end;
 i:=0;
 while a[i]=0 do i:=i+1;
 for j:=i to 25 do
 write(m[j]);
 writeln
end.
```

Пример правильной и эффективной программы на языке Бейсик:

```
DIM i, j, k, a(26) AS INTEGER
DIM s(26) AS STRING * 1
FOR i = 1 TO 26
a(i) = 0
s(i) = CHR$(ASC("a") + i - 1)
NEXT
INPUT c$
DO WHILE NOT (c$ = ".")
 a(ASC(c$) - ASC("a") + 1) = a(ASC(c$) - ASC("a") + 1) + 1
 INPUT c$
LOOP
```

```
FOR j = 1 TO 25
FOR i = 1 TO 25
  IF a(i) > a(i + 1) THEN
    k = a(i)
    c$ = s(i)
    a(i) = a(i + 1)
    s(i) = s(i + 1)
    a(i + 1) = k
    s(i + 1) = c$
 ENDIF
NEXT i
NEXT j
i = 1
DO WHILE a(i) = 0
 i = i + 1
LOOP
FOR j = i TO 26
PRINT s(j);
NEXT j
END
```

Вариант 10

24. **1) $c = 0$, $x = 0$. Значение x может быть не указано.**

2) Лишняя часть:

не нужно вводить x с клавиатуры;

верно: `readln(c)`.

3) Возможная доработка:

```
readln(c);
if c>0 then
write('нет решений')
else
if c = 0 then
write('x=0')
else
write('x=',sqrt(-c),
' или x=',-sqrt(-c));
```

(могут быть и другие способы доработки).

25.

На языке Паскаль	На языке Бейсик

`max:=-20;`
`for i:=1 to N do`
`if (a[i]<0) and (a[i]>max) then`
`max:=a[i];`
`writeln(max);`
	`MAX = -20`
	`FOR I = 1 TO N`
	`IF A(I) < 0 AND A(I) > MAX THEN`
	`MAX = A(I)`
	`ENDIF`
	`NEXT I`
	`PRINT MAX`

На языке СИ	**На естественном языке**

`max=-20;`
`for (i=0; i<N; i++)`
`if (a[i]<0 && a[i]>max)`
`max=a[i];`
`printf("%d", max);`
	Записываем в переменную MAX начальное значение, равное -20. В цикле от первого элемента до тридцатого сравниваем элементы исходного массива с нулём. Если текущий элемент меньше 0, то сравниваем значение текущего элемента массива со значением переменной MAX. Если текущий элемент массива больше MAX, то записываем в MAX значение этого элемента массива. Переходим к следующему элементу.
	После завершения цикла выводим значение переменной MAX.

26. Выигрывает первый игрок. Своим первым ходом он должен удвоить количество камней в первой куче. Для доказательства рассмотрим неполное дерево игры после этого хода первого игрока.

Позиция после первого хода	1-й ход второго игрока	Выигрывающий ход первого игрока	Пояснение
	4,6	7,6	Первый игрок выигрывает после любого ответа второго игрока, удвоив число камней в самой большой куче
4,3	7,3	7,6	-"-
	8,3	16,3	Выигрыш первого игрока

Из таблицы видно, что при первом ходе (2,3) → (4,3) первый игрок выигрывает не позже, чем на третьем ходу при любом ответе второго игрока.

27. Программа читает все входные данные один раз, не запоминая их в массиве, размер которого соответствует числу входных данных N или максимальной цене (3000). Во время чтения данных определяются две минимальных цены и количество АЗС, продающих 92-й бензин по этим ценам. При печати результата проверяется, что у кого-то цена больше минимальной (вторая по минимальности цена существует), в этом случае искомая (искомые) АЗС — со второй по величине ценой, если это не так, то искомая (искомые) АЗС — все, продающие 92-й бензин.

Баллы начисляются только за программу, которая решает задачу хотя бы для одного частного случая (например, когда все АЗС продают бензин по различной цене, и 92-й бензин продают не менее двух АЗС).

Пример правильной и эффективной программы на языке Паскаль:

```
var c: char;
 i, k, N, b, min1, min2, cnt1, cnt2: integer;
 s,s1,s2: string;
begin
 min1:=3001;
 cnt1:=0;
 readln(N);
 for i:=1 to N do
 begin
 read(c);
 s:='';
 repeat
 s:=s+c;
 read(c);
 until c=' '; {считана компания}
 repeat
 s:=s+c;
 read(c);
 until c=' '; {улица добавлена к компании}
 readln(k,b);
 if k = 92 then
 if min1 > b then
 begin
 min2:=min1; cnt2:=cnt1; s2:=s1;
 min1:=b; cnt1:=1; s1:=s
 end else
 if min1 = b then cnt1:=cnt1+1 else
 if min2 > b then
 begin
 min2:=b; cnt2:=1; s2:=s
 end else
 if min2 = b then cnt2:=cnt2+1
 end;
 if cnt2>0 then
 if cnt2=1 then writeln(s2) else writeln(cnt2)
 else {все АЗС продают 92-й бензин по одной цене}
 if cnt1=1 then writeln(s1) else writeln(cnt1);
 writeln;
end.
```

Пример правильной программы на языке Бейсик:

```
DIM s AS STRING
DIM s1 AS STRING, s2 AS STRING
min1 = 3001
cnt1 = 0
INPUT n
FOR j = 1 TO n
LINE INPUT s
i = 0
DO
i = i + 1
c$ = MID$(s, i, 1)
LOOP WHILE c$ <> " "
DO
i = i + 1
c$ = MID$(s, i, 1)
LOOP WHILE c$ <> " "
DO
i = i + 1
c$ = MID$(s, i, 1)
LOOP WHILE c$ <> " "
m = VAL(MID$(s, i + 1, 2))
b = VAL(MID$(s, i + 4))
k = i - 1
s = LEFT$(s, k)
IF m = 92 THEN
IF min1 > b THEN
 min2 = min1: cnt2 = cnt1: s2 = s1
 min1 = b: cnt1 = 1: s1 = s
 ELSE
 IF min1 = b THEN
 cnt1 = cnt1 + 1
 ELSE
 IF min2 > b THEN
 min2 = b: cnt2 = 1: s2 = s
 ELSE
 IF min2 = b THEN cnt2 = cnt2 + 1
 ENDIF
 ENDIF
ENDIF
```

```
ENDIF
NEXT j
IF cnt2 > 0 THEN
 IF cnt2 = 1 THEN PRINT s2 ELSE PRINT cnt2
 ELSE
 IF cnt1 = 1 THEN PRINT s1 ELSE PRINT cnt1
ENDIF
END
```