Информатика. 10 класс
Решения и ответы

1 вариант

№	Ответ	Балл	Решение
1	11_7 или 13_7	20	Нам неизвестно, добавила ли программа двойку к числу или нет, поэтому проверим оба варианта. 1) Допустим, двойка не прибавлялась, программа сразу же получила на выходе число 101011110. Переведём число 101011110 в семеричную систему счисления. Это число 350 в десятичной системе, в семеричной оно выглядит как 1010. Этот вариант нам подходит, так как после первого запуска программы должно было получиться число в двоичной записи. Так как количество единиц чётное, то снова неизвестно, добавила ли программа двойку к числу или нет. Опять имеет два варианта. 1.1) Если она получила 1010 без прибавления, то это число 10 в десятичной, в семеричной выглядит как 13_7. Если число 1010 было получено добавлением двойки в двоичной записи к первоначальному числу, то отнимем эту двойку. Получим: 1010-10=1000. Переведем 1000_2 в десятичную систему – это 8. Теперь перейдём в семеричную систему: $8_{10}=11_7$. 2) Если число 101011110 было получено добавлением двойки в двоичной записи к первоначальному числу, то отнимем эту двойку. Получим: 101011110-10=101011100. Переведём число в десятичную систему счисления – это число 348. В семеричной системе оно выглядит как 1005. Однако такой ответ нельзя получить (ведь должно было получиться число в двоичной записи). Следовательно, может быть только два варианта числа, которое было введено в программу при первом запуске: 11 или 13, т.е. 11_7 или 13_7. Ответ 11_7 или 13_7.
2	Первый игрок	15	Очевидно, что если в одной из куч орехов n, а в другой ноль, или же в обеих кучах орехов ровно n, то игрок, которому досталась такая ситуация, сразу выигрывает. Значит, игрок проигрывает, когда в его ситуации он может перейти только к такой выигрышной для противника ситуации. Например, при числе орехов 2 и 1 любой ход приводит к проигрышу. Значит, приходить к такой позиции нельзя. Отойдём чуть повыше и переберём маленькие значения орехов. Ситуация 3 и 1, выигрышная, так как, забирая 1 орех из большей кучи, мы приведем противника к заведомо проигрышной ситуации 2-1. Ситуация 4 и 7 схожа с 1 и 2; она проигрышная. Несложно убедиться перебором, что при любом ходе игрока соперник сможет его снова поставить на заведомо проигрышную позицию. В случае числа орехов 51 и 54 выигрывает первый игрок, так как своим первым ходом забирает по 47 орехов из обеих кучи и

			оставляет для второго игрока позицию 4 и 7. Как бы ни ходил второй игрок, можно будет свести игру к позиции 1 и 2, которая заведомо проигрышная.
3	1276	10	Из условия задачи следует, что нам нужно найти максимальную сумму монет, которую может собрать пират, пройдя из **левой нижней** клетки в **правую верхнюю**. Исходные данные находятся в области A1:J10. Для поиска максимального значения будем работать с областью A12:J21, так как при расчетах будем использовать исходные значения монет в каждой клетке. В ячейку A21 напишем значение **=A10**. Для каждой ячейки левого столбца это будет сумма всех ячеек ниже от текущей. Внесем в ячейку A20 формулу **=A9+A21** и скопируем за маркер вверх до ячейки A12. Для каждой ячейки строки №21 это будет сумма всех ячеек левее от текущей. Внесем в ячейку B21 формулу **=A21+B10** и скопируем за маркер вправо до ячейки J21. Далее в ячейку B20 вставим формулу **=B9+МАКС(A20;B21)** и скопируем за маркер в ячейки B12:J20. Значение в ячейке J12 будет максимальной суммой монет, которую может собрать пират — 1276. Ответ: 1276
4	Написанная программа	25	Нужно найти наибольшее специальное число, не превосходящее число n. Под специальным числом – понимается число, которое имеет в качестве простых делителей только 2, 3 и 7. Конечно же, число может делиться на 1 и на само себя. Если такого числа нет, то выведите 0. Также сделаем допущение, что число может делиться на 2^i, 3^j и 7^k ($i \geq 1$, $j \geq 1$, $k \geq 1$). Обратите внимание, что значение n может быть больше, чем значение 32-битной целочисленной переменной, поэтому необходимо использовать 64-битные числа. Данную задачу можно решить несколькими способами. Рассмотрим три из них: 1) медленный – переборный способ с проверкой всех делителей. 2) переборный (чуть быстрее) - деление всех подряд. 3) перебор в цикле все чисел вида $2^i*3^j*7^k$. **Первый способ**: 1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз). 2) Для каждого текущего числа x перебираем все простые делители и проверяем, если его делителями будут только 2, 3 и 7. **Второй способ**: 1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз). 2) Для каждого текущего числа x выполняем в цикле деление на 2, 3 и 7, пока не получим в итоге 1. Если так, то число нам подходит.

			Третий способ:
			Переберем в цикле все числа вида $2^i*3^j*7^k$ ($i\geq1$, $j\geq1$, $k\geq1$). Из условия задачи $n\leq10^{17}$, поэтому $1\leq i\leq56$, $1\leq j\leq35$, $1\leq k\leq20$). Среди этих чисел выберем наибольшее.
			
			Пример программы на языке Python (третий способ):
			n=int(input())
			x=0
			for i in range(1,56):
			  for j in range(1,35):
			  for k in range(1,20):
			  if 2**i*3**j*7**k<=n and x<=2**i*3**j*7**k:
			  x=2**i*3**j*7**k
			print(x)
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

5	Написанная программа	30	Алгоритм решения задачи:
			1) Сначала считываем N, длину лески M и массив координат (это массив целых чисел).
			2) Затем сортируем массив координат.
			3) Нам понадобится еще один массив S – для хранения накопленных расстояний между домами (сумм разностей между координатами домов).
			3) После этого будем находить решение методом динамического программирования. От двух точек ответом будет разность их координат (S2 = A2-A1), от более чем двух – разность координат между последними домами (ибо последний домик так или иначе придётся соединять с предпоследним) плюс min(Sk-1, Sk-
			
			Код в псевдоалгоритмическом языке:
			Заводим числа N и M, массивы a и S
			Считываем N и M
			От i = 1 до N считываем в массив a
			Сорт(a, a+n);//возможна любая сортировка
			От i = 2 до N: если i: = 2, то S[2] = a[2]-a[1]
			иначе S[i] = a[i] - a[i-1] +min(S[i-1], S[i-2])
			Если S[N]>=M то вывести S[N]
			иначе 0
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

Информатика. 10 класс
Решения и ответы

2 вариант

№	Ответ	Балл	Решение
1	20_6	20	Нам неизвестно, отнимала ли программа двойку от числу или нет, поэтому проверим оба варианта. 1) Допустим, двойка не отнималась, программа сразу же получила на выходе число 11011100. Переведём число 11011100 в шестеричную систему счисления. Это число 220 в десятичной системе, в шестеричной оно выглядит как 1004. Однако такой ответ нельзя получить, ведь после первого запуска программы должно было получиться число в двоичной записи. 2) Если число 11011100 было получено отниманием двойки в двоичной записи от первоначального числа, значит, добавим эту двойку. Получим: 11011100+10=11011110. Переведём число в десятичную систему счисления – это число 222 (т.е. 222_{10}). В шестеричной системе оно выглядит как 1010_6. Так как программа не могла выдать это число в качестве результата после первого запуска, ведь количество единиц в его записи чётно, значит, она получила его отниманием двойки в двоичной записи от первоначального числа. Тогда добавим эту двойку: 1010+10=1100. Переведем 1100_2 в десятичную систему – это 12. Теперь перейдём в шестеричную систему: $12_{10}=20_6$. Следовательно, при первом запуске в программу было введено число 20_6. Ответ 20_6.
2	Первый игрок	15	Очевидно, что если в одной из куч орехов n, а в другой ноль, или же в обеих кучах орехов ровно n, то игрок, которому досталась такая ситуация, сразу выигрывает. Значит, игрок проигрывает, когда в его ситуации он может перейти только к такой выигрышной для противника ситуации. Например, при числе орехов 2 и 1 любой ход приводит к проигрышу. Значит, приходить к такой позиции нельзя. Отойдём чуть повыше и переберём маленькие значения орехов. Ситуация 3 и 1, выигрышная, так как забирая 1 орех из большей кучи, мы приведем противника к заведомо проигрышной ситуации 2-1. Ситуация 4 и 7 схожа с 1 и 2; она проигрышная. Несложно убедиться перебором, что при любом ходе игрока соперник сможет его снова поставить на заведомо проигрышную позицию. Ситуация 3 и 5 также схожа с 1 и 2; несложно убедиться перебором, что при любом ходе соперник сможет его снова поставить на заведомо проигрышную позицию (2-1). В случае числа орехов 70 и 72 выигрывает первый игрок, так как своим первым ходом забирает по 67 орехов из обеих куч и оставляет для второго игрока позицию 3 и 5. Как бы ни ходил второй игрок, можно будет свести игру к позиции 1 и 2, которая заведомо проигрышная.

3	1057	10	Из условия задачи следует, что нам нужно найти максимальное количество моркови, которое может собрать Зайчик, пройдя из **левой верхней клетки** в **правую нижнюю**. Исходные данные находятся в области A1:J10. Для поиска максимального значения будем работать с областью A12:J21, так как при расчетах будем использовать исходные значения количества моркови в каждой клетке. В ячейку A12 напишем значение **=A1**. Для каждой ячейки левого столбца это будет сумма всех ячеек выше от текущей. Внесем в ячейку A13 формулу **=A2+A12** и скопируем за маркер вниз до ячейки A21. Для каждой ячейки строки №12 это будет сумма всех ячеек левее от текущей. Внесем в ячейку B12 формулу **=A12+B1** и скопируем за маркер вправо до ячейки J12. Далее в ячейку B13 вставим формулу = **=B2+МАКС(A13;B12)** и скопируем за маркер в ячейки B13:J21. Значение в ячейке J21 будет максимальным количеством моркови, которое может собрать Зайчик — 1057. Ответ: 1057
4	Написанная программа	25	Нужно найти наибольшее специальное число, не превосходящее число n. Под специальным числом – понимается число, которые имеют в качестве простых делителей только 3, 5 и 7. Конечно же, число может делиться на 1 и на само себя. Если такого числа нет, то выведите 0. Также сделаем допущение, что число может делиться на 3^i, 5^j и 7^k ($i \geq 1$, $j \geq 1$, $k \geq 1$). Обратите внимание, что значение n может быть больше, чем значение 32-битной целочисленной переменной, поэтому необходимо использовать 64-битные числа. Данную задачу можно решить несколькими способами. Рассмотрим три из них: 1) медленный – переборный способ. 2) переборный (чуть быстрее) - деление всех подряд. 3) перебор в цикле все чисел вида $3^i*5^j*7^k$. **Первый способ**: 1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз). 2) Для каждого текущего числа x перебираем все простые делители и проверяем, если его делителями будут только 3, 5 и 7. **Второй способ**: 1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз). 2) Для каждого текущего числа x выполняем в цикле деление на 3, 5 и 7, пока не получим в итоге 1. Если так, то число нам подходит. **Третий способ**: Переберем в цикле все числа вида $3^i*5^j*7^k$ ($i \geq 1$, $j \geq 1$, $k \geq 1$). Из условия задачи $n \leq 10^{17}$, поэтому $0 \leq i \leq 35$, $0 \leq j \leq 24$, $0 \leq k \leq 20$). Среди этих чисел выберем наибольшее.

			Пример программы на языке Python (третий способ):
			n=int(input())
			x=0
			for i in range(1,56):
			  for j in range(1,35):
			  for k in range(1,20):
			  if 3**i*5**j*7**k<=n and x<=3**i*5**j*7**k:
			  x=3**i*5**j*7**k
			print(x)
			
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

5	Написанная программа	30	Алгоритм решения задачи:
			1) Сначала считываем N, длину лески M и массив координат (это массив целых чисел).
			2) Затем сортируем массив координат.
			3) Нам понадобится еще один массив S – для хранения накопленных расстояний между домами (сумм разностей между координатами домов).
			3) После этого будем находить решение методом динамического программирования. От двух точек ответом будет разность их координат (S2 = A2-A1), от более чем двух – разность координат между последними домами (ибо последний домик так или иначе придётся соединять с предпоследним) плюс min(Sk-1, Sk-
			
			Код в псевдоалгоритмическом языке:
			Заводим числа N и M, массивы a и S
			Считываем N и M
			От i = 1 до N считываем в массив a
			Сорт(a, a+n);//возможна любая сортировка
			От i = 2 до N: если i: = 2, то S[2] = a[2]-a[1]
			иначе S[i] = a[i] - a[i-1] +min(S[i-1], S[i-2])
			Вывести S[N]
			
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

Информатика. 10 класс
Решения и ответы

3 вариант

№	Ответ	Балл	Решение
1.	20_5	20	Нам неизвестно, добавила ли программа двойку к числу или нет, поэтому проверим оба варианта. 1) Допустим, двойка не прибавлялась, программа сразу же получила на выходе число 10011000. Переведём число 10011000 в пятеричную систему счисления. Это число 152 в десятичной системе, в пятеричной оно выглядит как 1102. Однако такой ответ нельзя получить, ведь после первого запуска программы должно было получиться число в двоичной записи. 2) Если число 10011000 было получено добавлением двойки в двоичной записи к первоначальному числу, то отнимем эту двойку. Получим: 10011000-10=10010110. Переведём число в десятичную систему счисления – это число 150_{10}. В пятеричной системе оно выглядит как 1100. Так как программа не могла выдать это число в качестве результата после первого запуска, ведь количество единиц в его записи чётно, значит, она получила его прибавлением двойки в двоичной записи от первоначального числа. Тогда отнимем эту двойку: 1100-10=1010. Переведем 1010_2 в десятичную систему – это 10. Теперь перейдём в пятеричную систему: $10_{10}=20_5$. Следовательно, при первом запуске в программу было введено число 20_5. Ответ 20_5.
2.	Первый игрок	15	Очевидно, что если в одной из куч орехов n, а в другой ноль, или же в обеих кучах орехов ровно n, то игрок, которому досталась такая ситуация, сразу выигрывает. Значит, игрок проигрывает, когда в его ситуации он может перейти только к такой выигрышной для противника ситуации. Например, при числе орехов 2 и 1 любой ход приводит к проигрышу. Значит, приходить к такой позиции нельзя. Отойдём чуть повыше и переберём маленькие значения орехов. Ситуация 3 и 1, выигрышная для текущего игрока, так как, забирая 1 орех из большей кучи, мы приведем противника к заведомо проигрышной ситуации 2-1. Ситуация 4 и 7 схожа с 1 и 2; она проигрышная. Несложно убедиться перебором, что при любом ходе игрока соперник сможет его снова поставить на заведомо проигрышную позицию 1-2 или равного количества орехов.

			В случае числа орехов 77 и 80 выигрывает первый игрок, так как своим первым ходом забирает по 73 ореха из обеих кучи и оставляет для второго игрока позицию 4 и 7. Как бы ни ходил второй игрок, можно будет свести игру к позиции 1 и 2, которая заведомо проигрышная.
3.	152	10	Из условия задачи следует, что нам нужно найти максимальное количество образцов, которое может собрать марсоход, пройдя из **правой верхней клетки** в **левую нижнюю,** при заданном ограничении: можно собрать из каждой ячейки не более 8 образцов, т.е. если в ячейке находится число больше 8, то взять из нее мы можем только 8. Исходные данные находятся в области A1:J10. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(A1>8;8;A1) – это делаем для каждой ячейки. В измененной таблице все значения>8 заменятся на 8. Пусть измененная таблица хранится в области: A12:J21. Для поиска максимального значения будем работать с областью A23:J32, при расчетах будем использовать исходные значения количества образцов в каждой клетке из области A12:J21. В ячейку J23 напишем значение **=J12**. Для каждой ячейки правого столбца (J) это будет сумма всех ячеек выше от текущей. Внесем в ячейку J14 формулу **=J23+J13** и скопируем за маркер вниз до ячейки J23. Для каждой ячейки строки №23 это будет сумма всех ячеек правее от текущей. Внесем в ячейку I23 формулу **=J23+I12** и скопируем за маркер влево до ячейки A23. Далее в ячейку I24 вставим формулу **=I13+МАКС(I23;J24)** и скопируем за маркер в ячейки A24:J32. Значение в ячейке A32 будет максимальным количество образцов, которое может собрать марсоход при заданных условиях — 152. Ответ: 152
4.	Написанная программа	25	Нужно найти наибольшее специальное число, не превосходящее число n. Под специальным числом – понимается число, которые имеют в качестве простых делителей только 2, 5 и 7. Конечно же, число может делиться на 1 и на само себя. Если такого числа нет, то выведите 0. Также сделаем допущение, что число может делиться на 2^i, 5^j и 7^k ($i \geq 1$, $j \geq 1$, $k \geq 1$). Обратите внимание, что значение n может быть больше, чем значение 32-битной целочисленной переменной, поэтому необходимо использовать 64-битные числа. Данную задачу можно решить несколькими способами. Рассмотрим три из них: 1) медленный – переборный способ. 2) переборный (чуть быстрее) - деление всех подряд.

			3) перебор в цикле все чисел вида $2^i*5^j*7^k$.
			
			Первый способ:
			1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз).
			2) Для каждого текущего числа x перебираем все простые делители и проверяем, если его делителями будут только 2, 5 и 7.
			
			Второй способ:
			1) Перебираем в цикле все числа x в диапазоне от n до 1 (идем вниз).
			2) Для каждого текущего числа x выполняем в цикле деление на 2, 5 и 7, пока не получим в итоге 1. Если так, то число нам подходит.
			
			Третий способ:
			Переберем в цикле все числа вида $2^i*5^j*7^k$ ($i\geq1$, $j\geq1$, $k\geq1$). Из условия задачи $n\leq10^{17}$, поэтому $1\leq i\leq56$, $1\leq j\leq24$, $1\leq k\leq20$). Среди этих чисел выберем наибольшее.
			
			Пример программы на языке Python (третий способ):
			`n=int(input())`
			`x=0`
			`for i in range(1,56):`
			`  for j in range(1,24):`
			`  for k in range(1,20):`
			`  if 2**i*5**j*7**k<=n and x<=2**i*5**j*7**k:`
			`  x=2**i*5**j*7**k`
			`print(x)`
			
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

5.	Написанная программа	30	Алгоритм решения задачи: 1) Сначала считываем N, длину лески M и массив координат (это массив целых чисел). 2) Затем сортируем массив координат. 3) Нам понадобится еще один массив S – для хранения накопленных расстояний между столбами (сумм разностей между координатами столбов). 3) После этого будем находить решение методом динамического программирования. От двух точек ответом будет разность их координат (S2 = A2-A1), от более чем двух – разность координат между последними столбами (ибо последний столб так или иначе придётся соединять с предпоследним) плюс min(Sk-1, Sk-

			Код в псевдоалгоритмическом языке:
			Заводим числа N и M, массивы a и S
			Считываем N и M
			От i = 1 до N считываем в массив a
			Сорт(a, a+n);//возможна любая сортировка
			От i = 2 до N: если i: = 2, то S[2] = a[2]-a[1]
			иначе S[i] = a[i] - a[i-1] +min(S[i-1], S[i-2])
			Если S[N]>=M то вывести S[N]
			иначе 0
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

Информатика. 10 класс
Решения и ответы

4 вариант

№	Ответ	Балл	Решение
1.	1023 и 2046	20	Посчитаем количество букв А. Первая строка содержит одну букву А: $x_1 = 1$. Во второй строке вся первая строка удваивается и добавляется еще буква А, следовательно $x_2 = 2*x_1+1$. Аналогично для третьей строки: $x_3 = 2*x_2+1 = 2^2*x_1+2+1$. Следовательно, в строке с номером n количество букв А определяется по формуле: $$x_n = 2^{n-1}x_1 + \sum_{k=0}^{n-2} 2^k$$ Второе слагаемое – это сумма геометрической прогрессии с a1=1 и знаменателем q=2, следовательно $$\sum_{k=0}^{n-2} 2^k = 2^{n-1} - 1$$ $$x_n = 2^n - 1$$ Для n=10, x_{10}=1023. Аналогично рассуждая, можно получить формулу для общего количества символов: $y_n = 2^{n+1} - 2$. Для n=10, $y_{10} = 2^{11}-2=2046$. Ответ: 1023, 2046.
2.	49995	15	Очевидно, что получится девять четырёхзначных чисел, и независимо от того, как цифры были расставлены по кругу, каждая цифра по одному разу по участвует в записи единиц, по одному разу в записи десятков, по одному разу в записи сотен и по одному разу в записи тысяч. Таким образом, сумму всех полученных чисел можно было найти так: 9999+8888+7777+6666+5555+4444+3333+2222+1111=49995.
3.	256	10	Из условия задачи следует, что нам нужно найти максимальное количество шишек, которое может собрать Бельчонок, пройдя из **заданной** клетки в **правую верхнюю,** при заданном ограничении: можно собрать из каждой ячейки не более 12 шишек, т.е. если в ячейке находится число больше 12, то взять из нее мы можем только 12. Исходные данные находятся в области A1:U25. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(A1>12;12;A1) – это делаем для каждой

			ячейки. В измененной таблице все значения>12 заменятся на 12. Пусть измененная таблица хранится в области: A27:U51. Отметим следующее: точка начала сбора находится в ячейке K25, в измененной таблице это будет ячейка K51. Область левее столбца K нас не интересует, так как идти влево по условию задачи нельзя. Значит, будем работать с с областью K27:U51. Для поиска максимального значения будем работать с областью K53:U77, и при расчетах будем использовать значения образцов в каждой клетке (из области K53:U77). В ячейку K77 напишем значение **=K51**. Для каждой ячейки левого столбца это будет сумма всех ячеек выше от текущей. Внесем в ячейку A13 формулу **=K77+K50** и скопируем за маркер вниз до ячейки K53. Для каждой ячейки строки №77 это будет сумма всех ячеек левее от текущей. Внесем в ячейку L77 формулу =K77+L51 и скопируем за маркер вправо до ячейки U77. Далее в ячейку L76 вставим формулу **= =L50+МАКС(K76;L77)** и скопируем за маркер в ячейки L53:U76. Значение в ячейке U53 будет максимальным количествоv шишек, которое может собрать Бельчонок — 256. Ответ: 256
4.	Написан-ная програм ма	25	Условия задачи звучат так: пусть имеется многоугольник, который задан координатами своих N вершин в декартовой системе координат на плоскости: (x_1, y_1), (x_2, y_2), …, (x_N, y_N). Также даны координаты отрезка AB. Определить сколько сторон многоугольника пересекает отрезок AB. Предполагается, что отрезок не лежит полностью ни на одной стороне многоугольника. Если отрезок не пересекает ни одну из сторон, то программа должна вывести 0. Алгоритм решения: В этой задаче необходимо проверить пересекается ли заданный отрезок AB с каждым из N отрезков многоугольника C_iC_{i+1}. Задача облегчается предположением, что AB не лежит на границах многоугольника, следовательно, не надо отдельно проверять случай совпадения прямых. Пусть даны два отрезка 1) *AB* с координатами $A(x_1, y_1)$ и $B(x_2, y_2)$; 2) *CD* с координатами $C(x_3, y_3)$ и $D(x_4, y_4)$. Составляем уравнения прямых, образуемых отрезками *AB* и *CD*. Решаем полученную систему уравнений. Если система имеет единственное решение, следовательно это точка пересечения прямых. Однако, если пересекаются прямые, из

			этого еще не следует, что пересекаются отрезки. Дополнительно проверяем принадлежит ли точка пересечению обеим отрезкам. Составим уравнение прямой, проходящей через вершины *A* и *B*. *B*оспользуемся следующим известным уравнением $\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1}$ и преобразуем его к виду $a_1x+b_1y=c_1$, где $a_1=y_2-y_1$, $b_1=x_1-x_2$, $c_1=x_1(y_2-y_1)-y_1(x_2-x_1)$. Подставляем в него координаты концов второго отрезка. Если выполняются равенства: $$a_1x_3+b_1y_3=c_1 \text{ и } a_1x_4+b_1y_4=c_1$$ то обе точки *C* и *D* лежат на прямой *AB*. Это означает, что оба отрезка лежат на одной прямой. Дополнительно проверяем, имеют ли отрезки общую часть. Для этого проверяем выполняется ли следующее соотношение: $\min(x_3, x_4) \le \max(x_1, x_2)$ **and** $(\min(y_3, y_4) \le \max(y_1, y_2)$ Если отрезки не лежат на одной прямой, то проверяем имеют ли прямые, построенные продолжением этих отрезков точку пересечения. Составляем уравнение прямой, проходящей чрез вершины *C* и *D*: $a_2x+b_2y=c_2$, где $a_2=y_4-y_3$, $b_2=x_3-x_4$, $c_2=x_3(y_4-y_3)-y_3(x_4-x_3)$. Находим решение системы уравнений: $$\begin{cases}a_1x+b_1y=c_1\\a_2x+b_2y=c_2\end{cases}$$ Любым методом, например - методом Крамера. Вычисляем определитель системы: $$\Delta=\begin{vmatrix}a_1 & b_1\\a_2 & b_2\end{vmatrix}=a_1b_2-a_2b_1$$ Если определитель системы равен нулю, то прямые не пересекаются (следовательно, отрезки тоже не пересекаются). Если определитель не равен нулю, то находим точку пересечения прямых и проверяем, находится ли она внутри отрезков. Решение системы уравнений: $$x=\frac{1}{\Delta}\begin{vmatrix}c_1 & b_1\\c_2 & b_2\end{vmatrix}=\frac{1}{\Delta}(c_1b_2-c_2b_1)$$ $$y=\frac{1}{\Delta}\begin{vmatrix}a_1 & c_1\\a_2 & c_2\end{vmatrix}=\frac{1}{\Delta}(a_1c_2-a_2c_1)$$ Проверяем принадлежность точки пересечения отрезку *AB*: $x \ge \min(x_1, x_2)$ **and** $x \le \max(x_1, x_2)$ **and** $y \ge \min(y_1, y_2)$ **and** $y \le \max(y_1, y_2)$ и проверяем принадлежность точки пересечения отрезку *CD:*

			$x \geq \min(x_3, x_4)$ **and** $x \leq \max(x_3, x_4)$ **and** $y \geq \min(y_3, y_4)$ **and** $y \leq \max(y_3, y_4)$
			
			Программа на языке Delphi (среда разработки Code Gear 2009, консольное приложение):
			program Task_10_4_4;
			{$APPTYPE CONSOLE}
			Uses SysUtils, windows;
			var
			Xa, Ya, Xb, Yb:integer;
			N,i:integer;
			a1, b1, c1, a2, b2, c2, d, x, y:real;
			x1, y1, x2, y2, x0, y0:integer;
			count:integer=0;
			
			function min(a,b:real):real;
			begin
			if a<b **then** min:=a **else** min:=b;
			end;
			
			function max(a,b:real):real;
			begin
			if a>b **then** max:=a **else** max:=b;
			end;
			
			begin
			SetConsoleCP(1251);
			SetConsoleOutputCP(1251);
			write('Введите координаты точки A: ');readln(Xa, Ya);
			write('Введите координаты точки B: ');readln(Xb, Yb);
			a1:=Yb-Ya; b1:=Xa-Xb; c1:=Xa*(Yb-Ya)-Ya*(Xb-Xa);
			write('Введите количество вершин многоугольника: ');
			readln(N);
			write('Введите координаты 1-ой вершины: ');readln(x1,y1);
			X0:=X1; Y0:=Y1;
			for i := 1 **to** N-1 **do begin**
			write('Введите координаты ',i:2,'-ой вершины:
			');readln(x2,y2);
			a2:=Y2-Y1; b2:=X1-X2; c2:=X1*(Y2-Y1)-Y1*(X2-X1);
			d:=a1*b2-b1*a2;
			if d<>0 **then begin**
			x:=(c1*b2-b1*c2)/d;
			y:=(a1*c2-c1*a2)/d;

```
     if (x>=min(Xa,Xb)) and (x<=max(Xa,Xb)) and
(Y>=min(Ya,Yb)) and (y<=max(Ya,Yb)) then
       if (x>=min(X1,X2)) and (x<=max(X1,X2)) and
(Y>=min(Y1,Y2)) and (y<=max(Y1,Y2)) then
          count:=count+1;
   end;
   X1:=X2; Y1:=Y2;
  end;

  a2:=Y2-Y0; b2:=X0-X2; c2:=X0*(Y2-Y0)-Y0*(X2-X0);
  d:=a1*b2-b1*a2;
  if d<>0 then begin
   x:=(c1*b2-b1*c2)/d;
   y:=(a1*c2-c1*a2)/d;
   if (x>=min(Xa,Xb)) and (x<=max(Xa,Xb)) and
(Y>=min(Ya,Yb)) and (y<=max(Ya,Yb)) then
     if (x>=min(X0,X2)) and (x<=max(X0,X2)) and
(Y>=min(Y0,Y2)) and (y<=max(Y0,Y2)) then
       count:=count+1;
  end;
  if count>0 then writeln('Отрезок пересекается с ',count,'
сторонами многоугольника')
  else writeln('Отрезок не пересекается с многоугольником');
  readln;
end.
```

5.	Написанная программа	30	Бельчонок и Ёжик решили сыграть в математическую игру онлайн. Они отправляли друг другу положительные целые числа, не превышающие 1000. Всего количество отправленных чисел составило *N* штук (2<*N*<=*100*). Игра заключается в том, что необходимо расставить между числами знаки «+» и «-» так, чтобы значение получившегося выражения было равно заданному целому числу S. Если требуемый результат получить невозможно, то вывести «Решения нет», если можно – то вывести равенство на экран. Если решение не единственное, вывести любое. Помогите друзьям и напишите программу, которая решит эту задачу. **Примечание:** перед первым числом знак не ставится (оно всегда берется со знаком «+»). *Пример 1*. Числа 15, 25, 30 и результат 10. Решение 15+25-30=10. *Пример 2*. Числа 20 и 10, результат 100. Решения нет. **Решение**

			Считаем исходные значения в массив X с 1-го по *N*-й элемент. Знаки операций будем хранить в массиве логических значений sign, в котором *i*-ый элемент соответствует знаку после *i*-го числа (false соответствует знаку «+», true – означает знак «-»). Следует обратить внимание на то, что по условию задачи, перед первым слагаемым знака быть не может. Задача состоит в том, чтобы заполнить массив sign всему возможными способами и подсчитать суммы для этих способов. Если какая-нибудь из этих сумм будет равна *S*, то нужно вывести соответствующее выражение и завершить работу программы, а если ни одна не будет равна *S*, вывести сообщение об этом. Получить все возможные последовательности из плюсов и минусов можно, используя двоичную систему счисления: минус сопоставить нулю, плюс – единице. Двоичные представления чисел от 0 до 2^N-1, дополненные слева нулями до *N*-1 двоичных разрядов, дают все возможные последовательности длиной *N* из нулей и единиц. Будем получать комбинации в естественном порядке: сначала соответствующую 0, затем 1, затем 2 и так далее до 2^{N-1}-1. Пусть известно двоичное представление числа *X*. Как по этому представлению найти двоичную запись числа *X*+1? Если последняя цифра числа 0, то она увеличивается на единицу. Если последняя цифра числа 1, то эту цифру изменяем на 1 и повторяем алгоритм для следующей цифры. То есть, чтобы получить следующее двоичное число, нужно все концевые единицы заменить нулями, а первый с конца ноль заменить единицей, например: 1010 → 1011 100011 → 100100 111 → 1000 Сначала все элементы массива sign инициализируем false, это соответствует знаку «+» и двоичному нулю. Затем получаем комбинацию, соответствующую двоичной единице, - sign[1]=true, в остальных ячейках находятся false, т.к. sign[1] соответствует младшему двоичному разряду. Когда нужно прекратить перебор? Когда очередная полученная двоичная комбинация имеет больше *N*-1 разряда (между *N* числами необходимо расставить *N*-1 знака). Признаком этого является условие sign[N]=true. Фиктивный N-ый разряд sign[N] (в решении его нет) существует специально для того, чтобы узнать, когда заканчивать перебор. Для каждой комбинации значений массива sign вычисляем сумму S1:

```
S1:=X[1];
for i := 1 to N-1 do  if sign[i] then S1:=S1-X[i] else
S1:=S1+X[i];
```

Первоначально присваиваем сумме первое число. Далее в цикле к сумме либо прибавляем (если sign[i]=false), либо вычитаем (если sign[i]=true) следующее число.

Получившуюся сумму сравниваем с введенной суммой *S*. Если они не равны, то меняем комбинацию значений элементов массива sign и снова вычисляем сумму. Эти действия повторяем в цикле до тех пор, пока либо не переберем все комбинации (признак – sign[N]=true), либо сумма S1 станет равной сумме S.

Далее анализируем полученный результат. Если sign[N]=false, следовательно было получено нужное выражение. Выводим его на экран:

```
 for i := 1 to N-1 do begin
   write(X[i]:2);
   if sign[i] then write(' - ') else write(' + ');
 end;
 write(X[N],' = ',S);
```

Если sign[N]=true, значит все возможные комбинации были рассмотрены и выводим сообщение о том, что решения не существует.

Программа на языке Delphi (среда разработки Code Gear 2009, консольное приложение):

```
program Task_10_4_5;
{$APPTYPE CONSOLE}
uses  SysUtils, Windows;
const  MAX=100;
var
  N, i, S, S1:integer;
  X: array[1..MAX] of integer;       {Массив чисел}
  sign: array[1..MAX] of boolean;  {Массив знаков: false – знак «+», true – знак «-»}
begin
  SetConsoleCP(1251);
  SetConsoleOutputCP(1251);
  writeln('Выражение');

 {Ввод исходных данных:}
  readln(N); {Количество чисел последовательности N}
  readln(S); {Сумма S}
  for i := 1 to N do begin
    readln(X[i]);
```

```
  end;

 {Инициализируем исходную комбинацию знаков:
  все «+», находим первоначальную сумму S1}
  S1:=X[1]; sign[1]:=false;
  for i := 2 to N do begin
    S1:=S1+X[i];
    sign[i]:=false;
  end;

  {Цикл итераций}
  while (S1<>S) and not sign[N] do begin
   {Задаем новую комбинацию знаков}
   i:=1;
    while (i<N) and sign[i] do begin
      sign[i]:=not sign[i];
      i:=i+1;
    end;
    inc(sign[i]);

   {Вычисляем значение выражения с новой комбинацией
знаков}
    S1:=X[1];
    for i := 1 to N-1 do  if sign[i] then S1:=S1-X[i] else
S1:=S1+X[i];
  end;

  {Анализируем полученный результат}
  if not sign[N] then begin
    for i := 1 to N-1 do begin
      write(X[i]:2);
      if sign[i] then write(' - ') else write(' + ');
    end;
    write(X[N],' = ',S);
  end
  else writeln('Решения нет.');
  readln;
end.
```

Информатика. 10 класс
Критерии оценивания

Задача 1.

Правильный ответ с полным объяснением – 20 баллов.
Верное рассуждение, ответ частично неверный вследствие арифметических или логических ошибок – за каждое несовпадение отнимается 4 балла.
Верное рассуждение, но неполное, ответ верный – 16 баллов
Сделана только половина задания, состоящего из двух частей (причем выполнено верно) – 10 баллов.
Верное рассуждение, ответ неверный – 8 баллов.
Частично верное рассуждение, ответ неверный – 7 баллов.
Неверное рассуждение, неправильный ответ – 5 баллов.
Правильный ответ без пояснения – 3 балла.
Другой ответ – 0 баллов.

Задача 2.

Правильный ответ с полным объяснением – 15 баллов.
Верное рассуждение, ответ неверный (из-за арифметических ошибок) – 12 баллов.
В целом верное рассуждение, разобраны не все варианты стратегий выигрыша – 12 баллов.
В целом верное рассуждение, но недостаточно подробное, ответ верный – 11 баллов.
Частично верное рассуждение, ответ получен верный – 9 баллов.
Частично верное рассуждение, ответ неверный – 7 баллов.
Частично верное рассуждение, ответа нет – 7 баллов.
Неполное рассуждение, стратегия не описана полно (или слишком расплывчато), ответ верен – 7 баллов.
Неверное рассуждение, ответ верный – 6 баллов.
Неверное рассуждение, неправильный ответ – 5 баллов.
Неверное рассуждение, ответа нет – 5 баллов.
Правильный ответ без пояснения – 3 балла.
Другой ответ – 0 баллов.

Задача 3.

Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами с использование формул и функций– 10 баллов.
Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но недовведенными до финальной стадии – 8 баллов
Неправильный ответ и предоставлен файл с электронной таблицей с верными расчётами – 8 баллов
Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но без использования формул и функций – 7 баллов
Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами (не учтено одно из ограничений задачи) – 6 баллов.

Правильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 5 баллов.
Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами – 5 баллов.
Неправильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 4 балла.
Правильный ответ и не предоставлен файл с электронной таблицей, есть программа с верными расчётами – 4 балла.
Неправильный ответ и не предоставлен файл с электронной таблицей, есть программа с неверными расчётами – 3 балла.
Правильный ответ и не предоставлен файл с электронной таблицей с расчётами – 2 балла.
Ответ близок к правильному и не предоставлен файл с электронной таблицей с расчётами – 1 балл.
Другой ответ – 0 баллов.
Решен не свой вариант, ответ и расчёты верные – 5 баллов.

Задача 4.

Правильно решающий задачу, работающий и эффективный программный код – 25 баллов.
Правильно решающий задачу, работающий и неэффективный программный код – 23 балла.
Работающий и эффективный программный код, но есть незначительные ошибки в работе алгоритма – 18 баллов
Программный код работает, но он неэффективный и есть незначительные ошибки в работе алгоритма (частично верный код) – 16 баллов
Программный код частично верный, но не работающий – 12 баллов
Программный код работающий, но большая часть алгоритма ошибочна – 10 баллов
Программный код работающий, но полностью ошибочный – 8 баллов
Программный код неверный и не работающий – 5 баллов
Программный код неверный и не дописан – 4 баллов
Описан алгоритм работы программы, но не написан программный код – 2 балла. Другой ответ – 0 баллов.

Задача 5.

Правильно решающий задачу, работающий и эффективный программный код – 30 баллов.
Правильно решающий задачу, работающий и неэффективный программный код – 28 баллов.
Работающий и эффективный программный код, есть незначительные ошибки в работе алгоритма – 25 баллов.
Работающий и неэффективный программный код, есть незначительные ошибки в работе алгоритма – 21 балл
Есть некритичная ошибка в алгоритме, при ее исправлении алгоритм работает, и он будет эффективным – 19 баллов
Программный код работает, частично верный код, есть значительные ошибки в работе алгоритма – 15 баллов

Программный код частично верный, но не работающий – 15 баллов
Программный код работающий, но полностью ошибочный – 10 баллов
Программный код работающий, но решающий другую задачу (включая другой вариант) – 9 баллов
Программный код неверный и не работающий – 6 баллов
Программный код не дописан – 4 балла
Описан алгоритм работы программы, но не написан программный код – 3 балла. Другой ответ – 0 баллов