I. Задания заключительного этапа олимпиады 2016-17 года

Заключительный этап 11 класса (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (3 балла)

[Огромное число]

Вася получил длинную последовательность из цифр следующим образом. Он брал подряд натуральные числа, начиная с 1, переводил их в четверичную систему счисления и записывал результаты перевода друг за другом. Вот начало этой последовательности:

123101112132021222330313233100…

Вася остановился только тогда, когда дописал в конец последовательности четверичную запись числа 1023_{10}.

Затем он представил, что это одно большое число, записанное в четверичной системе счисления, и перевел его в шестнадцатеричную систему счисления.

Определите, какая шестнадцатеричная цифра стоит в этом числе на 60-ой позиции, считая слева направо от начала числа, а затем какая шестнадцатеричная цифра стоит на 1000-ой позиции. В ответе укажите эти две шестнадцатеричные цифры в указанном порядке через пробел.

Ответ: B 3

Решение:

Обратим внимание, что 1023_{10} это 33333_4, то есть максимальное пятиразрядное четверичное число.

Определим, структуру получившейся последовательности четверичных цифр. Она будет включать в себя, рассматривая от начала:

3 однозначных четверичных числа, занимающих, соответственно позиции в четверичной записи с 1 по 3.

*3*4=12 двузначных четверичных чисел (на первой позиции могут быть цифры 1, 2 или 3, а на второй позиции цифры от 0 до 3), занимающие позиции в четверичной записи с 4 по 12*2+3=27.*

*3*4*4=48 трехзначных четверичных чисел (аналогично, на первой позиции цифры от 1 до 3, а на второй и третьей – от 0 до 3), занимающие позиции в четверичной записи с 28 по 48*3+27=171.*

*3*4*4*4=192 четырехзначных четверичных числа, занимающие позиции в четверичной записи с 172 по 192*4+171=939.*

*3*4*4*4*4=768 пятизначных четверичных чисел, занимающие позиции в четверичной записи с 940 по 768*5+939=4779.*

Тогда всего запись этого длинного числа в четверичной системе счисления содержит 4779 цифр. Нам важно отметить, что это нечетное количество цифр. Известно, что для перевода числа из четверичной системы счисления в шестнадцатеричную достаточно каждой паре четверичных цифр поставить в соответствие одну шестнадцатеричную цифру, но рассматривая пары справа налево с конца числа. То есть шестнадцатеричное число будет начинаться с цифры 1 и далее содержать еще 2389 цифр.

*Первый вопрос задачи про шестнадцатеричную цифру на 60-ой позиции. Очевидно, что она соответствует 60*2-2=118 и 60*2-1=119 цифрам в четверичной записи числа. Эти цифры относятся к диапазону, в котором в последовательность были записаны трехзначные четверичные числа. И указанные четверичные цифры занимают, соответственно 118-27=91 и 92 позиции в этом диапазоне. Поделим 91 на 3 с остатком. Получим, что это 30 и 1 в остатке. То есть это 1 разряд в 31-ом трехзначном четверичном числе. Трехзначные четверичные числа начинаются с $100_4=16_{10}$. Значит это число $16+31-1=46_{10}=232_4$. Следовательно, шестнадцатеричная цифра на 60-ой позиции образована четверичными цифрами 23_4 и равна B_{16}.*

*Второй вопрос задачи про шестнадцатеричную цифру на 1000-ой позиции. Очевидно, что она соответствует 1000*2-2=1998 и 1000*2-1=1999 цифрам в четверичной записи числа. Эти цифры относятся к диапазону, в котором в последовательность были записаны пятизначные четверичные числа. И указанные четверичные цифры занимают, соответственно 1998-939=1059 и 1060 позиции в этом диапазоне. Поделим 1059 на 5 с остатком. Получим, что это 211 и 4 в остатке. То есть это 4 разряд в 212-ом пятизначном четверичном числе. Пятизначные четверичные числа начинаются с $10000_4=256_{10}$. Значит это число $256+212-1=467_{10}=13103_4$. Следовательно, шестнадцатеричная цифра на 1000-ой позиции образована четверичными цифрами 03_4 и равна 3_{16}.*

Возможно программное решение этой задачи на языке программирования, поддерживающем длинные строки.

2. Кодирование информации. Объем информации (2 балла)

[Опорные кадры]

Вася разрабатывает прототип цифровой видеокамеры. Видеокамера снимает кадры размером 1920 пикселей по горизонтали и 1080 пикселей по вертикали. Цвета всех пикселей кодируются с использованием палитры из 2^{24} оттенков.

Изначально Вася предположил, что будет записывать в память каждый кадр независимо, как последовательность кодов цветов пикселей, используя для записи каждого кода минимально возможное одинаковое для всех пикселей количество бит. Никакой другой информации, кроме кодов цветов пикселей в память не записывается.

Затем Вася стал анализировать результаты и обнаружил, что если разбить последовательность кадров на идущие подряд наборы из 6 кадров, то относительно каждого такого набора справедлива следующая закономерность. Во втором кадре набора оказывается ровно 1 процент пикселей, цвета которых отличаются от цветов пикселей, находящихся на этих же позициях в первом кадре набора. В третьем кадре набора оказывается ровно 2 процента пикселей, цвета которых отличаются от цветов пикселей, находящихся на этих же позициях в первом кадре набора и так далее. То есть, в каждом последующем кадре набора оказывается на 1 процент больше чем в предыдущем кадре набора пикселей, цвета которых отличаются от цветов пикселей, находящихся на этих же позициях в первом кадре набора.

Тогда Вася решил использовать более сложный способ записи данных в память. Для каждого набора он стал сначала записывать опорный кадр – первый кадр набора – как последовательность кодов цветов всех пикселей этого кадра, используя для записи каждого кода минимально возможное одинаковое для всех пикселей количество бит. А затем для

каждого из оставшихся 5-ти кадров набора он стал записывать в память только разницу с опорным кадром. Для этого он закодировал все позиции пикселей в кадре и для каждого кадра набора, кроме опорного, стал записывать множество пар значений: код позиции пикселя, цвет которого изменился, по отношению к цвету пикселя на этой же позиции в опорном кадре и код нового цвета этого пикселя. Для записи каждого кода позиции пикселя Вася использует минимально возможное одинаковое для всех кодов позиций количество бит. Для записи кода цвета Вася использует такое же количество бит, как и для записи кода цвета в опорном кадре. Все получаемые коды Вася записывает подряд побитно.

Определите, какой выигрыш по памяти получает Вася при новом способе записи данных в памяти для одного набора из 6 кадров. Округлите этот выигрыш **в меньшую** сторону до целого числа МБайт и запишите в ответ.

Примечание 1МБайт=1024 КБайт, 1 КБайт=1024 байт.

Ответ: 27

Решение:

*Обозначим за A количество пикселей в одном кадре. A=1920*1080. Поскольку используется палитра из 2^{24} оттенков, для хранения кода цвета пикселя, при условии минимально возможного одинакового для всех пикселей количества бит, будет использоваться $\log_2(2^{24})=24$ бита. Тогда в изначальном варианте для хранения набора из 6 кадров потребуется A*24*6=144*A бит.*

*Во втором варианте требуемый объем памяти будет состоять из двух частей. Первая – объем памяти для хранения опорного кадра. Он составит 24*A бит, поскольку опорный кадр хранится также, как и любой кадр в изначальном варианте хранения. Вторая часть – память, расходуемая для хранения пар значений для пикселей, меняющих цвет, относительно опорного. Сначала определим, сколько всего таких пикселей в оставшихся пяти кадрах. В первом кадре таких пикселей будет 0,01*A, во втором – 0,02*A и т.д. Соответственно, всего в пяти кадрах будет (0,01+0,02+0,03+0,04+0,05)*A=0,15*A пикселей, для которых нужно хранить пары значений. Теперь определим, сколько бит необходимо для хранения одной пары. Для записи кода нового цвета, очевидно, потребуется 24 бита, поскольку палитра оттенков используется одна и та же. Для того, чтобы определить, сколько потребуется бит на хранение кода позиции изменившего цвет пикселя, при условии использования минимально возможного одинакового для всех кодов позиций количества бит, необходимо вычислить логарифм по основанию 2 от количества позиций пикселей, то есть от количества пикселей в кадре и округлить в большую сторону до целого числа. $Log_2(A)=Log_2(1920*1080)=20,98370619$. Округлив в большую сторону, получаем, что для хранения кода позиции пикселя требуется 21 бит. Следовательно, для хранения одной пары потребуется 24+21=45 бит.*

*Теперь мы можем посчитать сколько всего бит потребуется для хранения набора из 6 кадров во втором случае. Это будет 24*A + 0,15*A*45=30,75*A бит. Тогда разность в требуемых объемах памяти составит 144*A-30,75*A=113,25*A бит. Вычислим это значение, переведя его в МБайт. 113,25*1920*1080/8/1024/1024=27,995 МБайт. Округлив в меньшую сторону, получаем 27 МБайт.*

3. Основы логики (1 балл)
[Таблица истинности]

Дан фрагмент таблицы истинности логической функции F(A,B,C,D), зависящей от четырех аргументов A, B, C и D.

A	B	C	D	F
0	0	1	1	0
1	0	0	1	1
1	1	0	1	0

Известно, что эту функцию можно задать в виде следующего логического выражения:

(([…] A → […] B) → […] C) → […] D,

где вместо некоторых […] может быть подставлен оператор логического отрицания. Определите, перед какими аргументами должны стоять операторы логического отрицания, чтобы получившаяся функция соответствовала приведенной таблице истинности.

В качестве ответа приведите последовательность из четырех знаков «+» или «-» в которой знак «+» будет означать, что, перед соответствующим аргументом в выражение не будет стоять оператор логического отрицания, а знак «-» будет означать, что перед соответствующим аргументом в выражение будет стоять оператор логического отрицания. Например, ответ «++-+» будет соответствовать выражению ((A → B) → not C)→D

Ответ: +-+-

Решение:

Обратим внимание на первую строку таблицы. С учетом скобок, последней операцией будет импликация, в правой части которой переменная […]D. Импликация может давать ложное значение только в одном случае, когда из истинного значения следует ложное. Поскольку в первой и третьей строках таблицы при истинном значении D получается ложное значение функции, следовательно, перед D должно быть логическое отрицание.

Теперь обратим внимание на вторую строку таблицы. В ней при D=1 функция принимает истинное значение. Но это возможно, только если ([…]A → […]B) → […]C также будет принимать ложное значение. Что в свою очередь означает, что […]C должно принимать ложное значение, а значит перед переменной C нет логического отрицания.

Наконец, рассмотрим третью строку. Отметим, что мы уже знаем, что перед переменной D есть логическое отрицание, а перед переменной C – его нет. Тогда получается, что (([…] A → […] B) → 0) → 0 = 0. Следовательно, ([…] A → […] B) → 0 = 1. Но тогда […] A → […] B должно быть ложно на значениях A=1 и B=1. Следовательно, перед A не должно быть логического отрицания, а перед B необходимо поставить логическое отрицание.

Теперь мы имеем все необходимое, чтобы записать ответ в соответствии с требованиями: «+-+-».

4. Кодирование информации. Алгоритмы обработки кодированной информации (1 балл) [Обратная польская нотация]

Обратная польская нотация (ОПН) – использующаяся в некоторых языках программирования форма записи математических выражений, в которой операнды расположены перед знаками операций.

Запись и вычисление выражения в ОПН устроены следующим образом:

- Выражение является последовательностью операндов и знаков операций.
- Выражение читается слева направо.
- Если в выражении встречается операнд, он заносится в стек операндов.
- Если в выражении встречается знак операции, выполняется соответствующая операция над двумя последними операндами в стеке и эти два операнда заменяются на результат вычисления операции.
- Результатом вычисления выражения становится результат последней вычисленной операции.

Пример вычисления выражения: 7 2 3 + - 4 *

Шаг	Текущее выражение	Стек операндов
1	7 2 3 + - 4 *	7
2	2 3 + - 4 *	7 2
3	3 + - 4 *	7 2 3
4	+ - 4 *	7 5
5	- 4 *	2
6	4 *	2 4
7	*	8

Дано выражение, записанное в ОПН:

2 4 4 #1 #2 8 1 #1 #3

За символами “#1”, “#2” и “#3” скрываются операции сложения, вычитания и умножения (в выражении присутствую все три операции), но неизвестно, какая операция соответствует какому символу. Подберите такой вариант соответствия символов операциям, при котором значение вычисленного выражения будет максимальным, и вычислите это значение. В ответе укажите целое число.

Ответ: 14

Решение:

Всего может быть 6 перестановок из трех различных операций. Составим таблицу, в которой для каждой перестановки приведем выражение, записанное в ОПН и его последовательное вычисление:

#1=”+”, #2=””, #3=”-“:* *2 4 4 + * 8 1 + -* *2 8 * 8 1 + -* *16 8 1 + -* *16 9 -* *7*	*#1=”+”, #2=”-”, #3=”*“:* *2 4 4 + - 8 1 + ** *2 8 – 8 1 + ** *-6 8 1 + ** *-6 9 ** *-54*
#1=””, #2=”+”, #3=”-“:* *2 4 4 * + 8 1 * -* *2 16 + 8 1 * -* *18 8 1 * -* *18 8 –* *10*	*#1=”*”, #2=”-”, #3=”+“:* *2 4 4 * - 8 1 * +* *2 16 – 8 1 * +* *-14 8 1 * +* *-14 8 +* *-6*
#1=”-”, #2=””, #3=”+“:* *2 4 4 - * 8 1 - +* *2 0 * 8 1 - +* *0 8 1 - +* *0 7 +* *7*	*#1=”-”, #2=”+”, #3=”*“:* *2 4 4 - + 8 1 - ** *2 0 + 8 1 - ** *2 8 1 - ** *2 7 ** *14*

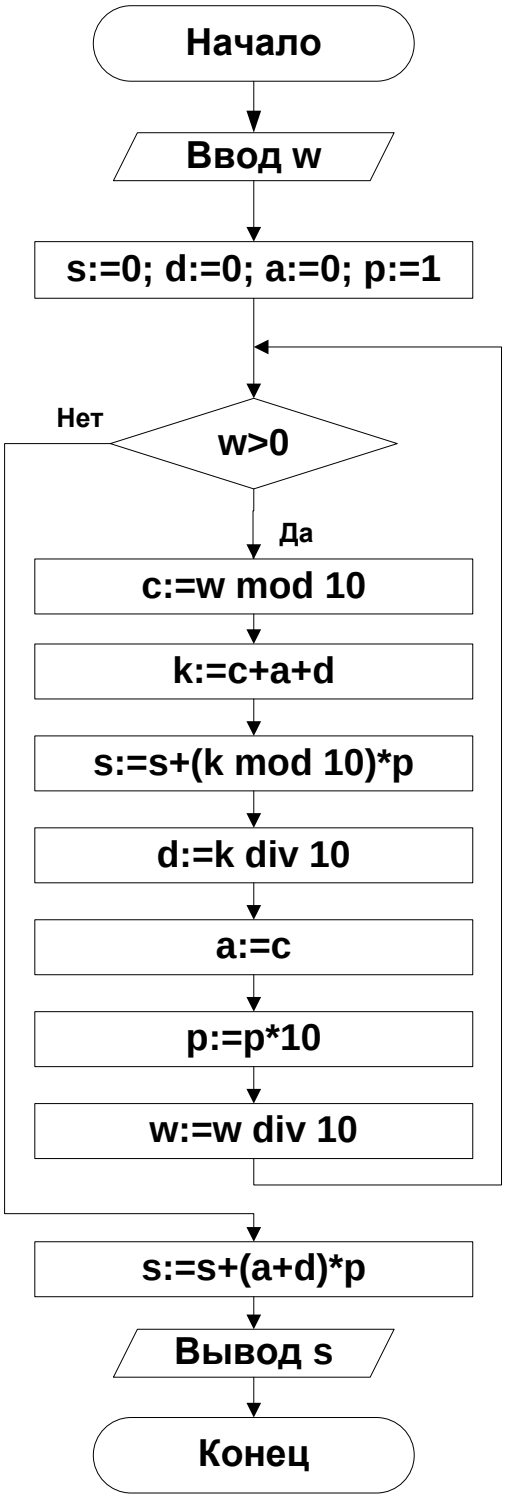
Из таблицы видно, что максимальное значение может быть получено в последнем случае, и оно составляет 14.

5. Алгоритмизация и программирование. Анализ алгоритма, заданного в виде блок-схемы (2 балла)

[Вычислитель]

Дана блок-схема алгоритма. Какое целое положительное число ***w*** необходимо подать на вход, чтобы после завершения алгоритма получилось значение ***s=96415***? В ответе укажите целое число.

*Примечание. Операция **mod** вычисляет остаток от деления первого аргумента на второй. Операция **div** вычисляет частное от целочисленного деления первого аргумента на второй.*



Ответ: 8765

Решение:

*Обратим внимание, что цикл будет выполняться столько раз, сколько цифр в записи числа **w**. Проанализируем значения переменных в конце очередной итерации цикла для нескольких вариантов значения **w** на входе:*

*Входное значение **w***	*Значения переменных в конце очередной итерации цикла*							*Выходное значение **s***
	c	***k***	***d***	***a***	***p***	***s***	***w***	
1	*1*	*1*	*0*	*1*	*10*	*1*	*0*	*11*
2	*2*	*2*	*0*	*2*	*10*	*2*	*0*	*22*
3	*3*	*3*	*0*	*3*	*10*	*3*	*0*	*33*
10	*0*	*0*	*0*	*0*	*10*	*0*	*1*	*110*
	1	*1*	*0*	*1*	*100*	*10*	*0*	
11	*1*	*1*	*0*	*1*	*10*	*1*	*1*	*121*
	1	*2*	*0*	*1*	*100*	*21*	*0*	
111	*1*	*1*	*0*	*1*	*10*	*1*	*11*	*1221*
	1	*2*	*0*	*1*	*100*	*21*	*1*	
	1	*2*	*0*	*1*	*1000*	*221*	*0*	

Проанализировав результаты, легко заметить, что алгоритм выполняет умножение входного значения на 11. Следовательно, для получения выходного значения ***s=96415****, на вход необходимо подать 96415/11=8765.*

6. Алгоритмизация и программирование. Формальные исполнители (2 балла) [Петляющий робот]

Робот движется по плоскости, с заданной на ней прямоугольной декартовой системой координат (ось X направлена слева направо, ось Y направлена снизу вверх), следующим образом:

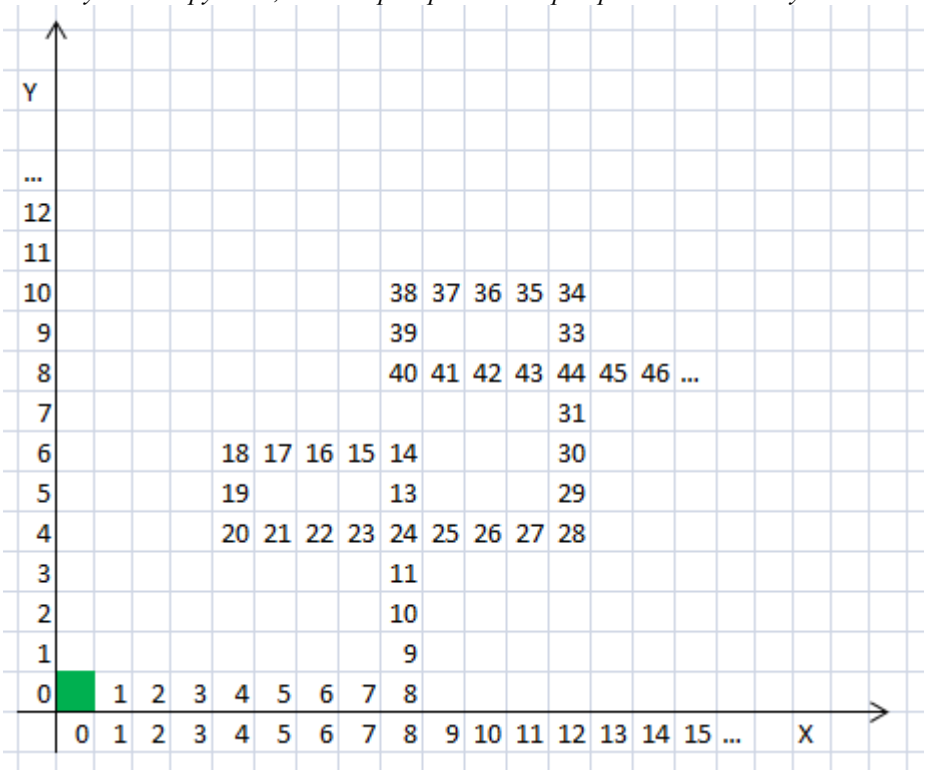
1. Робот может двигаться вперед или поворачиваться на 90 градусов против часовой стрелки.
2. Каждый ход робота – это изменение одной координаты на 1 в соответствие с текущим направлением его движения.
3. В памяти робота хранятся две переменных: A и B. После каждого хода робот увеличивает переменную B на 1 и делает проверку A=B.
4. Если проверка дает положительный результат, то робот выполняет следующие действия:
    a. повернуться на 90 градусов против часовой стрелки;
    b. обнулить переменную B.
    c. уменьшить переменную A на 2. Если после этого она стала равна 0, то записать в переменную A значение 8.
5. Описанные в пункте 4 действия не считаются отдельным ходом.

Робот начал движение вправо из точки с координатами (0,0). В момент начала движения переменная A=8, а переменная B=0. Робот остановился, сделав 1000 ходов. Определите, сколько существует точек на плоскости, в которых робот был два раза и координаты последней такой точки. В ответе укажите через пробел три целых числа: сначала количество точек на плоскости, в которых робот побывал два раза, затем координату X последней по пути следования робота точки, в которой он побывал два раза, а затем координату Y этой точки.

Ответ: 49 200 196

Решение:

Отметим, что робот попадает только в точки с целочисленными значениями обеих координат. Поэтому можем представить его движение в виде упрощенной схемы, приведенной ниже. Каждая клетка означает точку на плоскости с соответствующими координатами X и Y. Числа в клетках означают номер хода, завершив который робот оказывается в точке с такими координатами. Если робот попадает в определенную точку дважды, то число в соответствующей клетке соответствует номеру хода, на котором робот второй раз попал в точку с такими координатами.

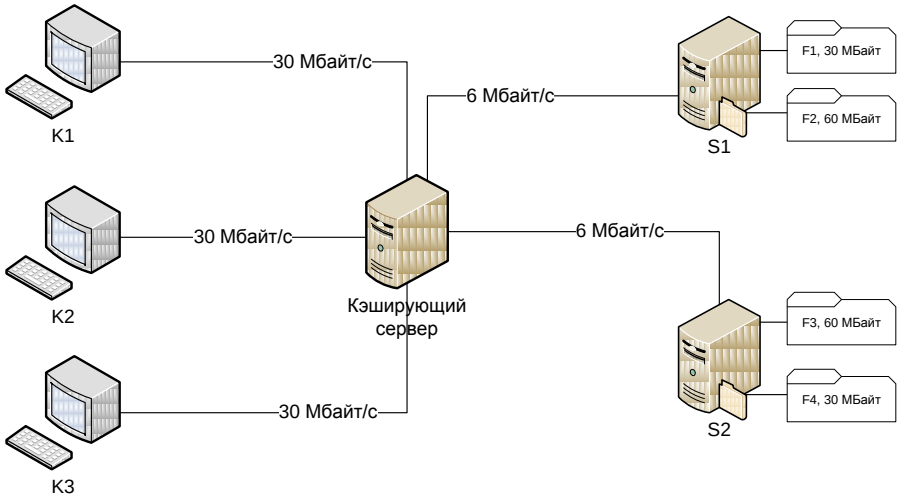


На схеме видно, что первая на пути робота точка, в которую он попадет дважды, имеет координаты (8,4) и робот повторно попадет в нее, завершив 24-ый ход. Следующая точка, в которую робот попадет дважды, будет иметь координаты (12,8) и робот повторно попадет в нее, завершив 44-ый ход. Легко показать, что у каждой последующей точки, в которую работ попадет дважды и координата X и координата Y будут на 4 больше, чем у предыдущей точки. При этом между повторным попаданием робота в эту точку и повторным попаданием робота в предыдущую точку робот сделает 20 ходов.

*Определим, сколько всего будет на пути робота точек, в которых он побывает дважды за 1000 ходов. Для этого осуществим целочисленное деление (1000-24):20 и прибавим 1 к результату. Получим 49. Следовательно, по пути робота будет 49 точек, которые он посетит дважды. Теперь найдем координаты последней по пути следования робота такой точки. Мы знаем координату первой точки, которую робот посетит дважды и то, как меняются координаты при переходе к следующей такой точке. Тогда координату X можно найти как 8+48*4=200 (к координате первой точки прибавляем 48 смещений на 4 к следующим точкам), а координату Y как 4+48*4=196. Следовательно, ответ задания: 49 200 196*

7. Телекоммуникационные технологии (2 балла)
[Кэширующий сервер]

Дана модель клиент-серверной системы с кэширующим сервером, приведенная на схеме:



Модель работает по следующим принципам.

1. На двух серверах данных S1 и S2 хранятся 4 файла с данными (F1 и F2 на сервере S1, F3 и F4 на сервере S2). Файлы F1 и F4 имеют размер по 30 МБайт каждый, а файлы F2 и F3 – по 60 МБайт каждый. Сервера данных связаны с кэширующим сервером каналами передачи данных со скоростями 6 МБайт/с. Кэширующий сервер связан с тремя клиентами каналами передачи данных со скоростями 30 МБайт/с.
2. Три клиента должны скачать все 4 файла каждый. У каждого клиента есть свой план, определяющий последовательность скачивания файлов. Клиенты в соответствие со своими планами присылают запросы на кэширующий сервер для скачивания конкретного файла. Клиент посылает запрос для скачивания следующего файла в момент, когда закончил скачивание предыдущего. Кэширующий сервер получает запрос мгновенно и сразу же начинает передачу данных клиенту из своей памяти или с сервера данных в соответствии с правилами, указанными ниже. Любой сервер может одновременно передавать данные нескольким клиентам.
3. Если в памяти кэширующего сервера **в момент поступления запроса** нет целиком нужного файла (в том числе, если он сейчас кэшируется, но кэширование не завершено), файл будет целиком скачан с сервера данных, на котором хранится этот файл. В этом случае скорость передачи данных клиенту равна скорости передачи данных в канале между сервером данных и кэширующим сервером. Скорость передачи данных между клиентом и кэширующим сервером в этом случае не учитывается.
4. Если передача некоторого файла запрашивается в первый раз, то кэширующий сервер начинает параллельно с передачей этого файла клиенту кэшировать его в свою память. Кэширование происходит с той же скоростью, что и передача файла клиенту и завершается в момент завершения передачи этого файла клиенту. Одновременные процессы кэширования нескольких файлов никак не влияют друг на друга. Процесс кэширования никак не влияет на процессы скачивания файлов. Кэширующий сервер имеет достаточно памяти для хранения всех четырех файлов и после кэширования хранит их сколь угодно долго.
5. Если в памяти кэширующего сервера **в момент поступления запроса** находится нужный файл, то он будет скачан с кэширующего сервера. Скорость скачивания клиентом файла с кэширующего сервера постоянная, зависит только от скорости передачи данных в канале между клиентом и кэширующим сервером и не зависит от количества одновременно выполняющихся операций скачивания данных клиентами или кэширования других файлов.
6. Если одновременно происходит передача нескольких файлов с одного сервера данных клиентам или передача одного и того же файла с сервера данных нескольким клиентам, то канал между сервером данных и кэширующим сервером на это время делится поровну на количество одновременных потоков передачи данных. Это значит, что скорость передачи уменьшается пропорционально количеству одновременных потоков получения данных. Даже если передается один и тот же файл одновременно нескольким клиентам, передача происходит независимыми потоками с разделением канала передачи данных.

Известно, что у клиентов были следующие планы скачивания файлов:

Клиент	Первый файл	Второй файл	Третий файл	Четвертый файл
K1	F1	F2	F3	F4
K2	F3	F4	F1	F2
K3	F1	F3	F2	F4

Все клиенты начали одновременно скачивать файлы в соответствии со своими планами. Определите время, через которое у всех клиентов будут скачаны все файлы. В ответе укажите целое число секунд.

Ответ: 40

Решение:

Рассмотрим события в модели в виде таблицы. В строках таблицы указано, что происходит в каждую секунду. В ячейках столбцов K1, K2 и K3 указано, какой файл скачивает в данную секунду соответствующий клиент.

Секунда	*К1*	*К2*	*К3*	*Содержимое кэша*	*Комментарии*
1	*F1*	*F3*	*F1*		*Канал передачи данных между S1 и кэширующим сервером поделен между двумя потоками скачивания файла F1, производится кэширование файлов F1 и F3*
2	*F1*	*F3*	*F1*		
3	*F1*	*F3*	*F1*		
4	*F1*	*F3*	*F1*		
5	*F1*	*F3*	*F1*		
6	*F1*	*F3*	*F1*		
7	*F1*	*F3*	*F1*		
8	*F1*	*F3*	*F1*		
9	*F1*	*F3*	*F1*		
10	*F1*	*F3*	*F1*		
11	*F2*	*F4*	*F3*	*F1, F3*	*F2 и F4 скачиваются независимо с разных серверов данных, файл F3 клиент К3 скачивает из кэша, производится кэширование файлов F2 и F4.*
12	*F2*	*F4*	*F3*	*F1, F3*	
13	*F2*	*F4*	*F2*	*F1, F3*	*Канал между S1 и кэширующим сервером поделен между двумя потоками скачивания файла F2, продолжается кэширование файлов F2 и F4. На начало 13-ой секунды клиенту К1 осталось скачать 48 МБайт от файла F2.*
14	*F2*	*F4*	*F2*	*F1, F3*	
15	*F2*	*F4*	*F2*	*F1, F3*	
16	*F2*	*F1*	*F2*	*F1, F3, F4*	*Продолжается параллельное скачивание файлов F2 клиентами К1 и К3. Клиент К2 скачивает файл F1 из кэша.*
17	*F2*	*F2*	*F2*	*F1, F3, F4*	*Канал между S1 и кэширующим сервером поделен между тремя потоками скачивания файла F2. На начало 17-ой секунды клиенту К1 остается скачать 36 МБайт файла F2, а клиенту К3 – 48 МБайт файла F2.*
18	*F2*	*F2*	*F2*	*F1, F3, F4*	
19	*F2*	*F2*	*F2*	*F1, F3, F4*	
20	*F2*	*F2*	*F2*	*F1, F3, F4*	
21	*F2*	*F2*	*F2*	*F1, F3, F4*	
22	*F2*	*F2*	*F2*	*F1, F3, F4*	
23	*F2*	*F2*	*F2*	*F1, F3, F4*	
24	*F2*	*F2*	*F2*	*F1, F3, F4*	
25	*F2*	*F2*	*F2*	*F1, F3, F4*	
26	*F2*	*F2*	*F2*	*F1, F3, F4*	
27	*F2*	*F2*	*F2*	*F1, F3, F4*	
28	*F2*	*F2*	*F2*	*F1, F3, F4*	
29	*F2*	*F2*	*F2*	*F1, F3, F4*	
30	*F2*	*F2*	*F2*	*F1, F3, F4*	
31	*F2*	*F2*	*F2*	*F1, F3, F4*	
32	*F2*	*F2*	*F2*	*F1, F3, F4*	
33	*F2*	*F2*	*F2*	*F1, F3, F4*	
34	*F2*	*F2*	*F2*	*F1, F3, F4*	
35	*F3*	*F2*	*F2*	*F1, F2, F3, F4*	*Канал между S1 и кэширующим сервером поделен между двумя потоками скачивания файла F2. На начало 35-ой секунды клиенту К2 остается скачать 24 МБайт файла F2, а клиенту К3 – 12 МБайт файла F2. Клиент К1 скачивает файл F3 из кэша.*
36	*F3*	*F2*	*F2*	*F1, F2, F3, F4*	
37	*F4*	*F2*	*F2*	*F1, F2, F3, F4*	*Продолжается параллельное скачивание файлов F2 клиентами К2 и К3. Клиент К4 скачивает файл F4 из кэша.*
38		*F2*	*F2*	*F1, F2, F3, F4*	*Продолжается параллельное скачивание файлов F2 клиентами К2 и К3.*
39		*F2*	*F4*	*F1, F2, F3, F4*	*На начало 39-ой секунды клиенту К2 осталось скачать 12 МБайт файла F2. Клиент К2 скачивает на максимальной скорости канала. Клиент К3 скачивает файл F4 из кэша.*
40		*F2*		*F1, F2, F3, F4*	*Клиент К2 завершает скачивание последнего файла.*

Как видно из таблицы, последний файл будет скачан клиентом К2 по прошествии 40 секунд от начала процесса скачивания файлов клиентами.

8. Технологии обработки информации в электронных таблицах (1 балл)

[Остатки]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	E
1					
2	0	=ОСТАТ($A2;B$1)		=СУММ(B2:C2)	
3	1				
4	2				
5	3				
6	4				
7	5				
8	6				
9	7				
10	8				

Диапазон ячеек A2:A100 заполнили последовательно по возрастанию целыми числами, начиная с 0. Ячейку B2 скопировали во все ячейки диапазона B2:C100. Ячейку D2 скопировали во все ячейки диапазона D3:D100. В ячейку B1 поместили число 15. Какое **минимальное** целое положительное число необходимо поместить в ячейку C1, чтобы в столбце D значение ячейки равное 1 первый раз (считая сверху вниз) встретилось в строке со значением ячейки в столбце A равном 45? В ответе укажите целое число.

Примечание. В англоязычной версии Microsoft Excel и в OpenOffice функции ОСТАТ соответствует функция MOD, а функции СУММ – функция SUM.

Ответ: 11

Решение:

После копирования ячейки B2 во все ячейки диапазона B2:C100 формулы в столбцах B и C будут вычислять в каждой строке остатки от деления чисел в ячейках столбца A на числа в ячейках B1 и C1 соответственно. Обратим внимание, что число 45 делится нацело на 15. Значит в ячейке столбца B этой строки, то есть в ячейке B47 значение 0. Следовательно, в ячейке C47 значение 1, а значит, число 44 делится нацело на значение в ячейке C1 и при этом оно больше 1. Соответственно потенциально в ячейке C1 могут быть значения 2, 4, 11, 22 или 44.

Но существуют дополнительные ограничения, поскольку нам необходимо, чтобы о рассмотренной выше строке значение 1 в столбце D появилось первый раз в этом столбце. Нужно рассмотреть ситуации, когда ранее в столбце B встречаются значения 0 или 1. Значение 0 в столбце B может встретиться соответственно при значении в столбце A=15 и при значении в столбце A=30. В этом случае, чтобы в столбце C не получилось значение 1, требуется, чтобы 15-1=14 или 30-1=29 не делились нацело на искомое число. Следовательно, нам не подходит значение 2.

Аналогично, значение 1 в столбце B может встретиться при значении в столбце A=16 и при значении в столбце A=31. В этом случае, чтобы в столбце C не получилось значение 0, требуется, чтобы и 16 и 31 не делились нацело на искомое число. Следовательно, нам не подходит значение 4. Тогда минимальным из оставшихся значений, удовлетворяющих всем ограничениям, является 11. Это и будет правильным ответом.

9. Технологии сортировки и фильтрации данных (3 балла)

[Поисковые запросы]

Даны 9 запросов к поисковому сервису некоторого сегмента сети Интернет:

1. диск & видеокарта | процессор
2. процессор & диск & видеокарта
3. процессор | память & видеокарта
4. процессор & память & диск & видеокарта
5. процессор & диск | память & процессор
6. процессор | память | диск
7. (память | диск) & процессор & видеокарта
8. видеокарта | процессор
9. видеокарта & память & процессор

В каждом запросе использованы ключевые слова и логические операции: “И” (обозначена “&”) и “ИЛИ” (обозначена “|”), приоритет логической операции “И” выше, чем приоритет логической операции “ИЛИ”. Во время выполнения всех 9 запросов сегмент сети оставался неизменным. В результате выполнения каждого запроса может быть выдано некоторое количество соответствующих ему страниц.

Требуется выбрать из предложенных ниже вариантов все те, которые соответствуют расположению номеров запросов в порядке невозрастания количества страниц, соответствующих запросу. Будем считать, что если для некоторой пары запросов невозможно однозначно определить порядок невозрастания количества страниц, соответствующих этим запросам, то они могут между собой следовать в любом порядке.

Варианты ответов:

1. 861357294
2. 681392574
3. 683152974

4. 631857294
5. 683157924
6. 681357924
7. 863175294
8. 683157294

Для доступа к ответам нажмите «Ответить».

Ответ: 1 5 6 8

Решение:

Рассмотрим запросы 6 и 8. Оба эти запроса содержат только операции ИЛИ. Отметим, что мы не можем однозначно расположить эти запросы в порядке невозрастания количества страниц, соответствующих запросу, поскольку мы не знаем, как соотносится между собой количество страниц, содержащих слова «память», «диск» и «видеокарта». При этом как мы покажем дальше все остальные запросы будут выдавать количество страниц однозначно не большее, чем любой из этих двух запросов. Обозначим конструкцией $\begin{bmatrix}6\\8\end{bmatrix}$ *факт, что на первой позиции в требуемой последовательности может быть или запрос 6 или запрос 8.*

Теперь рассмотрим запросы 1 и 3. Перепишем их, расположив слова немного в другом порядке для более удобного сравнения: «процессор | видеокарта & диск» и «процессор | видеокарта & память» соответственно. Обратим внимание, что мы не можем установить однозначно порядок невозрастания между ними. Сопоставим эти запросы с запросами из пары 6 и 8. Запрос «процессор | память | диск» однозначно даст не меньше страниц, чем любой из запросов «процессор | видеокарта & диск» или «процессор | видеокарта & память» поскольку множества страниц, которые будут получаться в ответ на любой из этих запросов, будут подмножествами множества страниц, получаемых по запросу «процессор | память | диск». Аналогично можно увидеть, что множества страниц, получаемых в результате выполнения запросов «процессор | видеокарта & диск» или «процессор | видеокарта & память» являются и подмножествами множества страниц, получаемых по запросу «видеокарта | процессор», а значит, последний запрос даст не меньше страниц, чем любой из пары 1 и 3. Тогда мы можем записать текущую обнаруженную последовательность так: $\begin{bmatrix}6\\8\end{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}$.

Теперь рассмотрим запрос 5 «процессор & диск | память & процессор». Перепишем его как «процессор & (диск | память) и обратим внимание, что множество страниц, получаемых в результате его выполнения, точно является подмножеством результата выполнения запроса «процессор», а значит и подмножеством результата выполнения любого из запросов 1 и 3. Следовательно, мы можем дополнить последовательность: $\begin{bmatrix}6\\8\end{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}5$.

Обратим внимание на запрос 7 «(память | диск) & процессор & видеокарта». Легко заметить, что страницы, получаемые в результате его выполнения, будут подмножеством страниц, получаемых в результате выполнения запроса 5 – добавляется требование, чтобы они содержали слово «видеокарта». Следовательно, наша последовательность приобретает вид: $\begin{bmatrix}6\\8\end{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}57$.

Теперь рассмотрим запросы 2 и 9, переписав их слова в удобном для сравнения порядке: «процессор & видеокарта & диск» и «процессор & видеокарта & память». Заметим, что между собой для этих запросов нельзя установить порядок невозрастания количества страниц, получающихся в результате их выполнения. Но сопоставим их с запросом 7: «(память | диск) & процессор & видеокарта». Этот запрос можно переписать следующим образом: «процессор & видеокарта & память | процессор & видеокарта & диск». Очевидно, что страницы, получаемые в результате выполнения запроса 2 или запроса 9, будут подмножествами множества страниц, получаемого в результате запроса 7. Значит, мы можем дополнить нашу последовательность: $\begin{bmatrix}6\\8\end{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}57\begin{bmatrix}2\\9\end{bmatrix}$.

Наконец, рассмотрим запрос 4: «процессор & память & диск & видеокарта». Легко заметить, что страницы, получаемые в результате его выполнения, будут подмножеством и множества страниц, получаемых в результате выполнения запроса 2 и множества страниц, получаемых в результате выполнения запроса 9. Следовательно, окончательная последовательность будет такой: $\begin{bmatrix}6\\8\end{bmatrix}\begin{bmatrix}1\\3\end{bmatrix}57\begin{bmatrix}2\\9\end{bmatrix}4$.

Теперь осталось выбрать из вариантов ответов те, которые совпадают с полученной последовательностью:

1. *861357294 – совпадает*
2. *681392574 – не совпадает*
3. *683152974 – не совпадает*
4. *631857294 – не совпадает*
5. *683157924 – совпадает*
6. *681357924 – совпадает*
7. *863175294 – не совпадает*
8. *683157294 – совпадает*

10. Технологии программирования (2 балла)

В Берляндии каждый автомобиль имеет регистрационный номер. Автомобильные номера в Берляндии имеют следующий вид: **LDDLDDL**, где символ **L** обозначает строчную латинскую букву, а **D** цифру.

Филипп устроился работать в службу регистрации автомобильных номеров. По своей неопытности в первый же день работы Филипп разлил на стопку номеров кофе. У некоторых номеров оказался залит второй блок цифр (цифры на позициях 5 и 6).

Филипп считает, что все номера в Берляндии уникальны, поэтому он хочет быстро подобрать все залитые цифры, так чтобы среди всех номеров не было двух одинаковых. Задача показалась ему нерешаемой, и он попросил вас помочь ему.

Формат входного файла

В первой строке входного файла **input.txt** записано натуральное число **n**, не превосходящее **1000** — количество номеров в стопке.

В следующих **n** строках находятся **n** регистрационных номеров, в **i+1**-й строке **i**-й номер, в описанном выше формате. На месте залитых цифр находятся знаки вопросов.

Гарантируется, что знаки вопроса могут находиться только на месте цифр из второго блока цифр, причем либо на позициях обеих цифр, либо ни на одной из позиций.

Формат выходного файла

Первая строка выходного файла **output.txt** должна содержать **NO**, если в стопке были одинаковые номера. Иначе первая строка должна содержать **YES**, а далее **n** строк должны содержать номера из стопки — по одному в каждой строке, причем **i+1**-я строка должна содержать **i**-й номер. Номера должны удовлетворять принятому в Берляндии формату в том же порядке, что и во входном файле.

Если ответов несколько — разрешается вывести любой.

Обратите внимание, что Филипп хочет восстановить только залитые цифры, то есть он должен заменить только знаки вопросов цифрами, другие символы в номерах измениться не должны.

Пример входных и выходных данных

input.txt	output.txt
4 a10a10c a30b??c a30b??c x70r??r	YES a10a10c a30b10c a30b22c x70r37r
3 a00b10c a00b10c c02y03x	NO
2 a99a??b a99a??b	YES a99a11b a99a22b

11. Технологии программирования (4 балла)

На уроке информатики учитель рассказал Васе про новый вид строк — *максимально-символьные* строки. Строка называется *максимально-символьной*, если символ, который встречается в ней максимальное количество раз, единственен. Например, строка **"abacaba"** — *максимально-символьная*, потому что единственный символ, который встречается максимальное количество раз в ней — '**a**'. В то же время строка **"cabacbac"** — не *максимально-символьная*, потому что символы '**a**' и '**c**' встречаются в ней максимальное количество раз, то есть не являются единственными.

После урока Вася сразу начал думать над следующей задачей: из данного набора символов составить как можно меньше *максимально-символьных* строк, используя все символы из набора ровно по одному разу в любом порядке. Вася не смог придумать решение этой задачи, поэтому обратился за помощью к вам. Помогите ему!

Формат входного файла

В единственной строке входного файла **input.txt** записана строка **s**, характеризующая набор символов. Ее длина не превосходит **100.**

Формат выходного файла

В первой строке выходного файла **output.txt** требуется вывести минимальное количество *максимально-символьных* строк **k**, которое можно составить из данного набора, использовав каждый символ ровно один раз.

В следующих **k** строках выходного файла требуется вывести *максимально-символьные* строки составленные из данного набора.

Если существует несколько правильных ответов, разрешается вывести любой из них.

Пример входных и выходных данных

input.txt	output.txt
abacaba	1 abacaba
abcabc	2 aab ccb
abc	3 a b c
cabacbac	2 bcb acaca