Отборочный этап 11 класса. 2 тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (2 балла)

[Остатки]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D	E
1			=B1+2		
2		=ОСТАТ($A2;B$1)			
3	=A2+1				
4					
5					
6					
7					

Ячейку ***C1*** скопировали в ячейки ***D1*** и ***E1***. Ячейку ***A3*** скопировали в ячейки ***A4*** и ***A5***. Ячейку ***B2*** скопировали во все ячейки диапазона ***B2:E5***. Определите, какие два **двузначных** натуральных числа записали в ячейки ***B1*** и ***A2***, если в результате вычисления формул получились значения в ячейках: ***B5=5***, ***C5=3***, ***D5=1***, ***E5=18***. В ответе запишите через пробел два натуральных числа: сначала число, которое было записано в ячейку ***B1***, а потом число, которое было записано в ячейку ***A2***.

2. Электронные таблицы. Графики и диаграммы (1 балл)

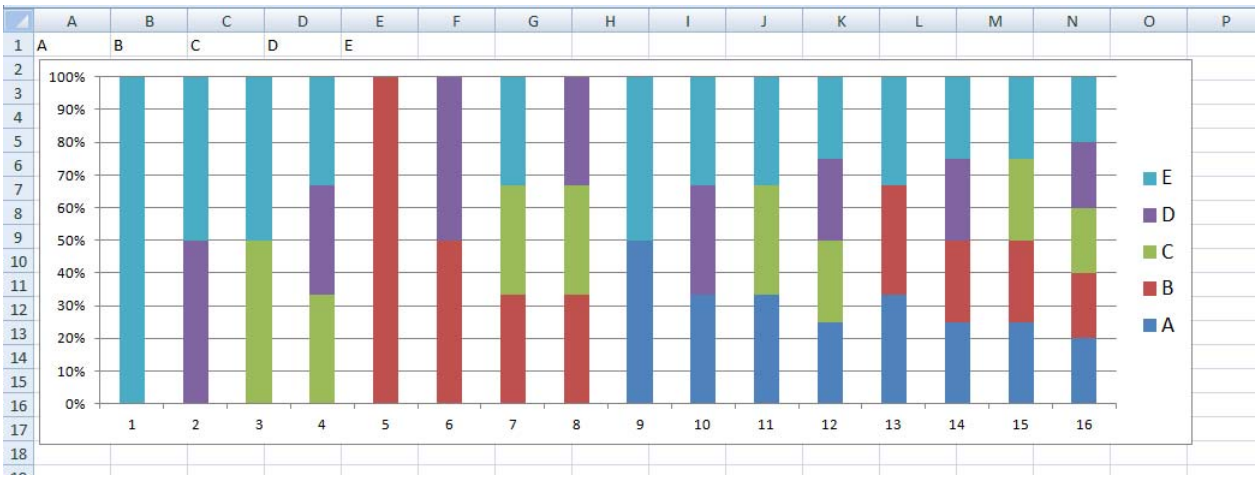
[Логичные столбики]

В ячейку E2 электронной таблицы поместили формулу:

=ЕСЛИ(ИЛИ(НЕ(И(#2=1;НЕ(#2=1)));И(НЕ(#2=1);#2=1));1;0),

где знаками # зашифрованы имена столбцов: A, B, C и D. Известно, что каждое из этих имен столбцов встречается в формуле ровно один раз.

В каждую ячейку диапазона A2:D17 записали или 0 или 1 (в результате часть ячеек диапазона содержит число 0, а часть – число 1), а ячейку E2 скопировали во все ячейки диапазона E3:E17. После этого выделили диапазон A2:E17 и построили для выделенного диапазона нормированную гистограмму с накоплением. Результат приведен на рисунке:



Определите, какая формула была записана в ячейке E2. В ответе укажите подряд без пробелов четыре буквы: названия столбцов в порядке их упоминания в формуле. Например, если построенная гистограмма свидетельствует о том, что в ячейке E2 была записана формула «=ЕСЛИ(ИЛИ(НЕ(И(D2=1;НЕ(C2=1)));И(НЕ(B2=1);A2=1));1;0)», то в ответ следует записать DCBA.

3. Сортировка и фильтрация данных (2 балла)

[Перемешанная таблица]

Одним из принципов сортировки символьных последовательностей является сортировка в лексикографическом порядке на основе сравнения кодов отдельных символов по определенной кодовой таблице. В кодовой таблице каждому символу взаимно однозначно соответствует некоторое неотрицательное целое число – код этого символа. В этом случае мы говорим, что последовательность символов ***α*** предшествует последовательности символов ***β***, если первые k символов (k>=0) обеих последовательностей совпадают, а k+1 символ последовательности ***α*** имеет код меньше, чем k+1 символ последовательности ***β***.

Арабские цифры тоже являются символами алфавита. В большинстве кодовых таблиц им сопоставляются идущие подряд последовательности кодов, так, чтобы код большей цифры оказывался больше кода меньшей цифры. Например, в таблице ASCII арабским цифрам соответствуют коды от 48 до 57, причем символу «0» соответствует код 48, символу «1» – код 49 и так до символа «9» с кодом 57. Поэтому если мы отсортируем по описанному выше принципу последовательности цифр одинаковой длины, то результат этой сортировки совпадет с сортировкой десятичных чисел, соответствующих этим последовательностям.

Снусмумрик решил подшутить над Муми-троллем и поменял на его компьютере кодовую таблицу таким образом, что арабские цифры по-прежнему занимают диапазон кодов **от 48 до 57**, но не расположены в порядке возрастания, а перемешаны. В результате, когда Муми-тролль отсортировал **по возрастанию** набор из 16-ти четырехсимвольных цифровых последовательностей, он получил такой результат:

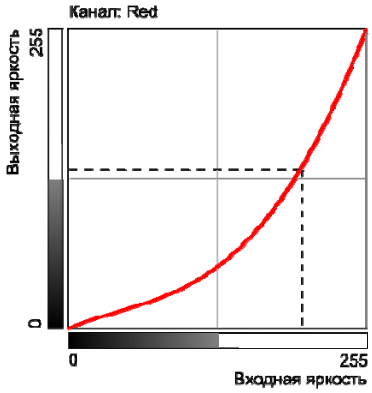
3520
3526
3588
3584
3136
3134
3105
3103
2919
2917
2990
2997
2875
2879
2826
2821

Определите, какие коды имеют арабские цифры в кодовой таблице, измененной Снусмумриком. В ответе укажите через пробел три числа: сначала код цифры 2, затем код цифры 7 и затем код цифры 9.

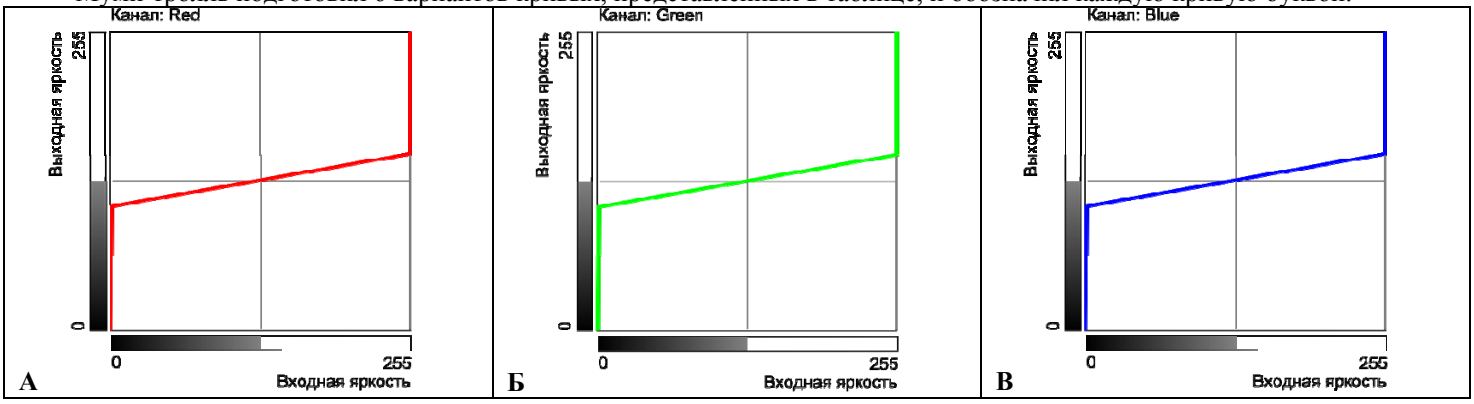
4. Мультимедиа технологии (3 балла)

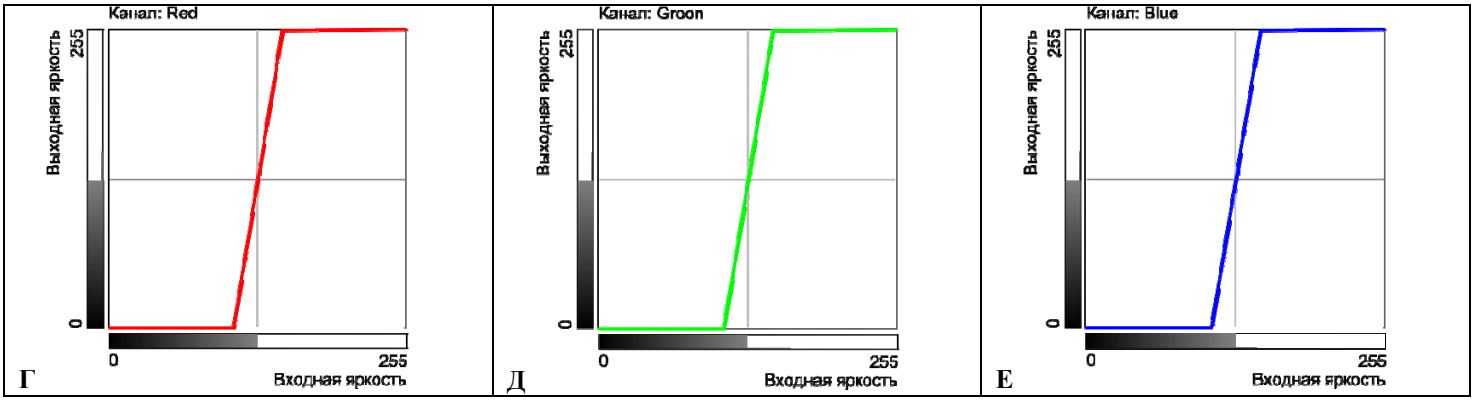
[Кривые]

Одним из распространенных инструментов цветокоррекции является инструмент «Кривые» («Curves»). Инструмент представляет собой график, показывающей соотношение входной и выходной яркости всего изображения или отдельного цветового канала (например, в модели RGB – красного, зеленого или синего). По горизонтальной оси откладываются значения яркости исходного изображения, а по вертикальной оси – нового изображения, получаемого в результате цветокоррекции. Таким образом, в зависимости от формы кривой для всех точек с одинаковой яркостью по соответствующему каналу устанавливаются новые значения яркости. Например, в соответствии с кривой, представленной ниже, все точки, имевшие по красному каналу яркость 200, теперь будут иметь по этому каналу яркость 135 и т.д.

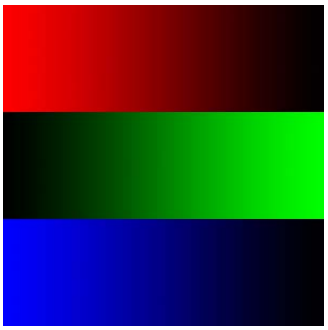


Муми-тролль подготовил 6 вариантов кривых, представленных в таблице, и обозначил каждую кривую буквой:

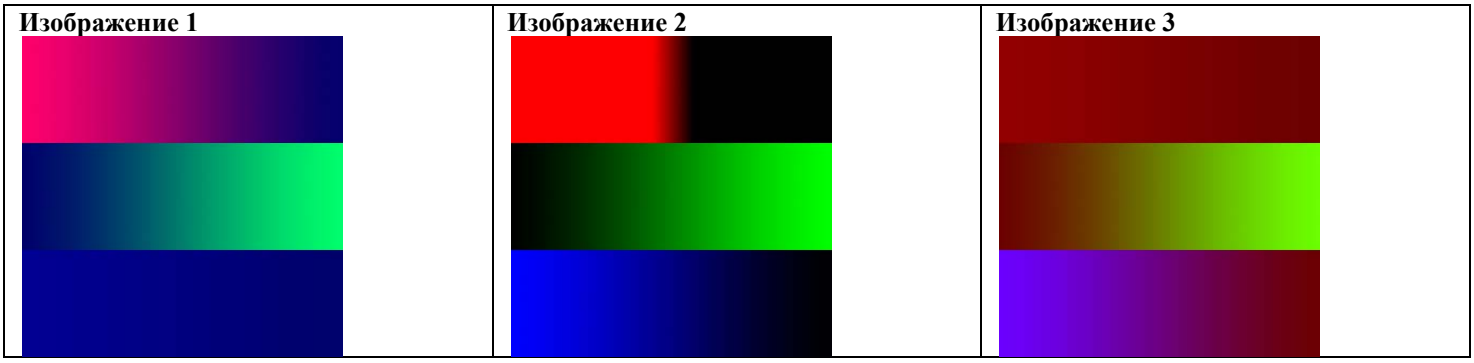




Муми-тролль решил проверить, как работают эти кривые. Он подготовил следующее исходное изображение:



Затем он опробовал на этом изображении три из подготовленных им кривых и получил следующие изображения:



Известно, что в каждом случае к исходному изображению была применена только одна из подготовленных кривых. Определите, какой из вариантов получен с помощью какой кривой. В ответе укажите подряд без пробелов последовательность из трех букв: сначала букву, соответствующую кривой, с помощью которой получено изображение 1, затем букву, соответствующую кривой, с помощью которой получено изображение 2, и затем букву, соответствующую кривой, с помощью которой получено изображение 3.

5. Телекоммуникационные технологии (3 балла). [Сниффер]

Большинство сетевых приложений построено по архитектуре клиент-сервер. Взаимодействие приложений через стек протоколов TCP\IP осуществляется с использованием многоуровневой адресации. Данные между компьютерами (хостами) доставляет протокол IP. Каждое сообщение протокола IP (IP-пакет) в заголовке содержит IP-адрес отправителя и IP-адрес получателя. По IP-адресу получателя определяется маршрут следования пакета и осуществляется его доставка. В IP-пакет инкапсулировано или сообщение протокола TCP, или сообщение UDP. Протоколы TCP и UDP доставляют сообщения конкретному приложению на хосте. В заголовках TCP и UDP содержатся особые адреса взаимодействующих приложений - номера порта отправителя и порта получателя. Таким образом, сообщение сначала доставляется на хост по IP-адресу получателя, а потом конкретному приложению по номеру порта получателя.

При взаимодействии клиента и сервера заранее известен номер порта программы-сервера. Это или стандартный для определенного прикладного протокола номер порта, или любой порт, на использование которого настроена программа-сервер. Например, для протокола HTTP (Web) стандартный порт сервера - 80 TCP, но Web-сервер может быть настроен на любой порт, положим – 8081 TCP. В первом случае для подключения к серверу на Web-клиенте достаточно указать адрес хоста, на котором работает программа-сервер, а номер порта получателя (по умолчанию 80 TCP) будет подставлен в заголовок TCP автоматически. Во втором случае в адресной строке после адреса надо явно указывать номер порта через двоеточие (например, 192.168.0.1:8081).

Программа-клиент занимает любой свободный порт больше 1024. На одном компьютере может быть запущено несколько клиентов, которые займут разные порты, при этом эти клиенты могут работать как с разными серверами, так и с одним и тем же сервером.

Для анализа трафика используются программы-снифферы. Они служат для перехвата всего трафика (входящего и исходящего) на наблюдаемом сетевом интерфейсе (например, сетевой карте) и последующего его анализа. Программы-снифферы имеют встроенный язык запросов для отбора части перехваченных сообщений по формальным признакам.

Студент Пётр изучает сетевые технологии. IP-адрес компьютера Петра 192.168.0.1. Пётр на своем компьютере запустил FTP-сервер в пассивном режиме и Web-сервер на стандартных портах (21 TCP и 80 TCP соответственно), к которым подключились другие студенты его группы. Пётр запустил на своем компьютере несколько Web-браузеров и FTP-клиентов и установил соединение с несколькими внешними Web и FTP-серверами, работающими по стандартным портам TCP.

Кроме этого Пётр запустил на своем компьютере SSH-клиент и подключился к SSH-серверу, который настроен на работу по порту 80 TCP.

Для анализа трафика Пётр использует программу-сниффер Wireshark.

Пётр осуществлял перехват трафика в течение 10 минут и решил провести анализ трафика. Для фильтрации трафика Пётр решил использовать только следующие условия:

http - отбирает все сообщения, содержащие данные и (или) команды протокола HTTP

ftp - отбирает все сообщения, содержащие данные и (или) команды протокола FTP

ssh - отбирает все сообщения, содержащие данные и (или) команды протокола SSH

ip.addr - отбирает все сообщения, содержащие IP пакеты, отправленные с подходящего по условию адреса или на него

ip.src - (от англ. source - источник) отбирает все сообщения, содержащие IP пакеты, отправленные с подходящего по условию адреса

ip.dst - (от англ. destination - адресат) отбирает все сообщения, содержащие IP пакеты, отправленные на подходящий по условию адрес

tcp.port - отбирает все сообщения, содержащие TCP сегменты, отправленные с подходящего по условию порта TCP или на него

tcp.srcport - отбирает все сообщения, содержащие TCP сегменты, отправленные с подходящего по условию порта TCP

tcp.dstport - отбирает все сообщения, содержащие TCP сегменты, отправленные на подходящий по условию порт TCP

Доступны логические операторы и операторы сравнения:

&& - И

|| - ИЛИ

== - равно

!= - НЕ равно

() - скобки для управления порядком вычисления условия

! – логическое НЕ (например выражение !http – отбирает все пакеты кроме содержащих данные и (или) команды протокола HTTP)

> - больше

< - меньше

>= - больше или равно

<= - меньше или равно

Выберите **все** выражения-фильтры в программе Wireshark, которые позволят Петру отобрать среди всего перехваченного трафика ***только*** сообщения Web-клиентов и FTP-клиентов (от них и к ним), запущенных на его компьютере. Работа по протоколам HTTP и FTP напрямую, без серверов посредников (proxy). Никакие другие программы, кроме указанных, не передают данные по TCP\IP. Для доступа к ответам нажмите «Ответить».

1. http && ftp
2. http && ((ip.src==192.168.0.1 && tcp.dstport==80) || (ip.dst==192.168.0.1 && tcp.srcport==80)) || ftp && ((ip.src==192.168.0.1 && tcp.dstport==21) || (ip.dst==192.168.0.1 && tcp.srcport==21))
3. (http || ftp) && ip.addr==192.168.0.1
4. (ip.src==192.168.0.1 && tcp.dstport==80) || (ip.dst==192.168.0.1 && tcp.srcport==80) || (ip.src==192.168.0.1 && tcp.dstport==21) || (ip.dst==192.168.0.1 && tcp.srcport==21)
5. http && ip.addr=192.168.0.1 && tcp.port=80 || ftp && ip.addr=192.168.0.1 && tcp.port=21
6. (http || ftp) && ((ip.src==192.168.0.1 && tcp.srcport>1024 || ip.dst==192.168.0.1 && tcp.dstport>1024))
7. (http || ftp) && (ip.addr==192.168.0.1 && (tcp.srcport>1024 || tcp.dstport>1024))
8. (http || ftp) && tcp.port>1024

6. Операционные системы (4 балла) [Round Robin]

В вычислительных системах с одним центральным процессором в каждый момент времени центральный процессор может осуществлять вычисления только одного из процессов. Многозадачные операционные системы, поддерживающие одновременно несколько процессов, разделяют ресурс центрального процессора между процессами, поочередно предоставляя каждому из процессов определенное время использования центрального процессора, тем самым реализуя

«псевдопараллельное» выполнение этих процессов. Правило, по которому ресурс распределяет время между несколькими потребителями, называется дисциплиной обслуживания.

Одной из распространенных дисциплин обслуживания для управления ресурсом центрального процессора в операционных системах является ***круговая очередь*** (Round Robin). Каждому процессу, находящемуся в этой очереди, присвоен порядковый номер. Операционная система сначала дает процессу с минимальным номером использовать центральный процессор в течение заданного периода времени, называемого ***квантом непрерывного выполнения***. По истечении этого кванта, операционная система предоставляет такой же по продолжительности квант непрерывного выполнения процессу со следующим по возрастанию номером. Если закончился квант непрерывного выполнения процесса с максимальным номером из этой очереди, следующий квант предоставляется опять процессу с минимальным номером и так далее. Если процесс завершил вычисления, он выбывает из очереди. При этом если он не использовал полностью последний квант непрерывного выполнения, операционная система не дожидается окончания кванта и предоставляет новый квант непрерывного выполнения следующему по круговой очереди процессу.

Продолжительность кванта непрерывного выполнения и продолжительность вычислений на центральном процессоре, необходимую каждому процессу, можно измерять в условных единицах – ***тактах***. Например, если процессу P_0 нужно до завершения периода вычислений 12 тактов, а квант непрерывного выполнения составляет 5 тактов, то в общей сложности квант непрерывного выполнения должен быть предоставлен такому процессу 3 раза, причем в последний раз он завершит вычисления до окончания выделенного кванта. Для ***передачи кванта непрерывного выполнения*** следующему процессу операционная система должна выполнить ряд служебных операций, затрачивая на них фиксированное количество тактов. Это количество тактов одинаково при любой операции передачи кванта непрерывного исполнения другому процессу, в том числе, если предыдущий процесс завершился, не использовав полностью свой квант непрерывного выполнения. Если передача кванта непрерывного выполнения происходит тому же процессу, который перед этим выполнялся, поскольку в очереди не осталось других процессов, операционная система не затрачивает на такую передачу дополнительных тактов и очередной квант непрерывного выполнения этого процесса начинается сразу же после окончания предыдущего.

Для оценки производительности часто используется показатель ***среднее время выполнения процессов***. Для каждого процесса определяется ***полное время выполнения процесса*** – время от его появления в очереди до завершения его вычисления в последнем предоставленном ему кванте непрерывного выполнения. Это время включает в себя как такты, на протяжении которых он вычислялся на центральном процессоре, так и такты, которые он ожидал своей очереди на вычисление. Времена полного выполнения всех процессов суммируются и делятся на количество процессов. В результате получается ***среднее время выполнения процессов***. Этот показатель зависит от выбранной продолжительности кванта непрерывного выполнения.

Задача. В начальный момент времени очередь образовали одновременно появившиеся три процесса, которым присвоены номера P_0, P_1 и P_2. При этом процессу P_0 требуется до завершения периода вычислений 59 тактов, процессу P_1 – 29 тактов, а процессу P_2 – 11 тактов.

Время, необходимое операционной системе на ***передачу кванта непрерывного выполнения*** очередному процессу составляет 2 такта. В начальный момент времени операционная система сразу же предоставляет квант непрерывного выполнения процессу P_0, не затрачивая дополнительных тактов на передачу кванта непрерывного выполнения.

Определите, при какой продолжительности ***кванта непрерывного выполнения***, ***среднее время выполнения*** этих процессов окажется минимальным. В ответе укажите целое число – количество тактов в этом кванте непрерывного выполнения.

7. Технологии программирования (3 балла)

Вы работаете бухгалтером в некоторой компании. В компании работает **n** человек. Про каждого человека известно, какую зарплату он получает. Также, иногда, компания выдаёт каждому сотруднику премию. Размер премии может варьироваться, но всем сотрудникам выдаётся премия одинакового размера. Кроме того, иногда нужно составлять финансовый отчёт, в котором нужно сравнить суммы, получаемые каждым сотрудником в качестве зарплат и в качестве премий.

Более конкретно, могут происходить следующие ситуации:

- Каждый работник получает свою зарплату $\mathbf{a_i}$.
- Каждый работник получает премию **b**. **b** может быть различным для разных выдач премии.
- Для отчёта, нужно сказать, сколько работников на данный момент получили больше премиями, чем зарплатой.

Формат входного файла

В первой строке входного файла **input.txt** находится целое число **n** ($\mathbf{1 \le n \le 500}$) — число работников компании. В следующей строке заданы **n** целых чисел $\mathbf{a_i}$ ($\mathbf{1 \le a_i \le 500}$) — зарплаты работников.

В третьей строке задано число запросов **t** ($\mathbf{1 \le t \le 500}$). Далее, в **t** строках идут сами запросы. Запросы бывают трёх видов:

9. «1» — выдать каждому работнику зарплату.

«2 **b**» ($\mathbf{1 \le b \le 500}$) — выдать каждому работнику премию **b**.

«3» — узнать, сколько работников получили больше премиями, чем зарплатами.

Формат выходного файла

Выведите в собственной строке ответ на каждый запрос типа «3».

Пример входных и выходных данных

input.txt	output.txt
3 10 20 30 5	1 3

1 2 20 3 2 20 3	

8. Технологии программирования (6 баллов)

Вы — директор завода по производству кваса. Ваш завод произвёл **n** литров кваса, который теперь нужно разлить по бутылкам. У вас есть достаточно большое количество бутылок трёх различных объёмов: **a**, **b** и **c** литров.

Вам не хочется терять ни литра драгоценного продукта, поэтому, все **n** литров должны быть разлиты по бутылкам. Также, вы хотите использовать как можно меньшее число бутылок. Определите, сколько бутылок каждого типа нужно наполнить.

Формат входного файла

В первой строке входного файла **input.txt** находится целое число **n** (**1 ≤ n ≤ 500**) — имеющееся число литров кваса. В следующей строке заданы целые числа **a**, **b** и **c** (**1 ≤ a, b, c ≤ 500**) — объёмы бутылок первого, второго и третьего вида, соответственно.

Формат выходного файла

В первой строке выходного файла **output.txt** требуется вывести «`YES`», если можно разлить все **n** литров по бутылкам заданного объёма, и «`NO`, если это сделать нельзя.

В случае, если в первой строке выведено «`YES`», во второй строке должно содержаться три числа: сколько бутылок первого, второго и третьего вида, соответственно, нужно наполнить. Сумма выведенных чисел должна быть минимальна среди всех возможных ответов.

В случае, если ответов с минимальным суммарным количеством бутылок несколько, выведите любой из них.

Пример входных и выходных данных

input.txt	output.txt
9 2 4 6	NO
14 3 6 2	YES 0 2 1