I. Задания заключительного этапа олимпиады 2013-14 года

Заключительный этап 11 класса (приведен один из вариантов заданий)

1. Кодирование информации. Системы счисления (2 балла) [Сдвиговый умножитель]

Незнайка сконструировал компьютер – умножитель. Компьютер имеет два регистра **A** и **B** по 24 двоичных разряда каждый и поддерживает следующую систему команд:

Команда	Действие
<N+	Осуществить сдвиг двоичной записи числа в регистре **A** на *N* бит влево (в освобождающиеся справа позиции записываются значения 0). Затем сложить значение, полученное в регистре **A** со значением, занесенным в регистр **B** и записать результат в регистр **B**. *N* – целое неотрицательное число. После завершения команды регистр **A** содержит результат сдвига.
M{…}	Выполнить *M* раз последовательность команд, заключенную в фигурные скобки. M не может превышать 10.

Написав программу для этого компьютера, можно выполнить операцию умножения целого положительного числа, помещаемого в регистр **A** на любое целое положительное число (если не возникнет переполнение одного из регистров в процессе вычислений). Результат по завершении программы будет находиться в регистре **B**. Легко заметить, что выполнив, например, программу **2{<0+<1+}<2+** можно получить в регистре **B** результат умножения числа, помещаемого в регистр **A** на 25_{10}. Также отметим, что такого же результата можно добиться, выполнив программу и с меньшим количеством символов, например **<0+<3+<1+**.

Известно, что в регистр **A** поместили число, не превосходящее 2560_{10}. Составьте и напишите в ответе программу, содержащую минимально возможное количество символов, которая позволит вычислить результат умножения числа, помещенного в регистр **A** на 6553_{10}. Перед началом выполнения программы значение регистра **B** равно нулю.

Ответ: <0+3{<3+<1+}

Решение

Обратим внимание, что операция сдвига двоичной записи числа в регистре A на N бит влево с записью нулей в освобождающиеся справа позиции эквивалентна умножению числа в регистре A на 2^N. Также отметим, что любое целое положительное число можно представить в виде суммы слагаемых $c_n*2^n+c_{n-1}*2^{n-1}+…+c_1*2+c_0$, где c_i – разряды двоичной записи этого числа могут принимать значения 0 или 1. Тогда умножение числа A на это число можно представить как $A*c_n*2^n+A*c_{n-1}*2^{n-1}+…+A*c_1*2+A*c_0$. Причем те слагаемые, в которых c_i равно нулю мы можем не учитывать, а слагаемые, в которых c_i равно единице эквивалентны $A*2^i$. Следовательно, можно умножение числа в регистре A на произвольное целое положительное число представить как последовательность операций сдвига и сложения получаемых результатов. Причем, поскольку по указанной системе команд после каждого сдвига в регистре A сохраняется результат этого сдвига, мы должны последовательно сдвигать число на количество разрядов, соответствующее количеству нулей, разделяющих очередные две единицы, рассматривая число справа налево, и суммировать результат с предыдущим после каждой операции сдвига. Переведем число 6553 в двоичную систему счисления. Результат будет равен: 1100110011001. Исходя из изложенных выше рассуждений, умножение на это число в указанном выше синтаксисе операций сдвига и сложения может быть записано как <0+<3+<1+<3+<1+<3+<1+. Теперь обратим внимание, что кроем этого система команд предполагает возможность организации цикла. Легко проверить, что использование цикла <0+3{<3+<1+} даст наиболее короткую запись программы, что и будет правильным ответом.

2. Кодирование информации. Объем информации (1 балл) [Одно из двух]

Винтик и Шпунтик разрабатывают систему удаленной черно-белой печати текста. На вход клиентского приложения системы подается страница, содержащая N символов текста – последовательность из N кодов символов (каждый символ кодируется одинаковым минимально возможным количеством бит). На выходе серверного приложения должно быть сформировано растровое изображение страницы, составленное из изображений отдельных символов, которое потом будет передано на принтер. Поскольку используется моноширинный шрифт одного размера, все изображения отдельных символов имеют одинаковый размер. Самой трудоемкой операцией при этом является растрирование – построение растрового изображения символа, и Винтик со Шпунтиком не могут решить, выполнять ли эту операцию в клиентском приложении или в серверном.

Они рассматривают два варианта алгоритмов работы системы:

Алгоритм **А**:

1. В клиентском приложении последовательно преобразовать каждый символ страницы в его растровое изображение, получив N изображений, размером 32 на 64 пикселей каждое. Изображения кодируются без сжатия. Палитра цветов любого изображения содержит 2 цвета;
2. После того, как сформированы растровые изображения *всех* N символов, передать все полученные изображения по каналу передачи данных серверному приложению. Передается только N несжатых изображений символов в порядке их следования в тексте. Никакой дополнительной информации в канале не передается;
3. После получения всех N растровых изображений символов, в серверном приложении сформировать из растровых изображений отдельных букв растровое изображение всей страницы.

Алгоритм **Б**:

1. Не осуществляя никакой обработки в клиентском приложении, передать коды символов по каналу передачи данных. Передается только N кодов символов. Никакой дополнительной информации в канале не передается;

2. После получения *всех* N кодов символов, в серверном приложении последовательно преобразовать каждый полученный символ в его растровое изображение, получив N изображений;
3. После получения всех N растровых изображений символов, в серверном приложении сформировать из растровых изображений отдельных букв растровое изображение всей страницы.

Известно, что количество символов на странице N = 2048. При наборе текста использовался алфавит из 128 символов. Скорость передачи данных в канале равна 1024 бит в секунду. Растрирование одного символа занимает 1 секунду, если она выполняется в клиентском приложении, и 3 секунды, если она выполняется в серверном приложении. Пункты 3 в обоих алгоритмах выполняются за одинаковое время.

Определите, какой из предложенных алгоритмов будет эффективнее в указанных условиях и насколько. В ответе укажите сначала букву А или букву Б, в зависимости от того на выполнение какого алгоритма потребуется меньше времени, а затем через пробел целое число, показывающее на сколько больше секунд потребуется на выполнение другого алгоритма.

Примечание. Любыми другими временными затратами, кроме явно указанных в условии, следует пренебречь.

Ответ: А 14

Решение

Рассчитаем время выполнения каждого алгоритма без учета времени на пункты 3 алгоритмов, поскольку их выполнение занимает одинаковое время и не будет влиять на разность времен выполнения алгоритмов.

*T_A = 2048 * 1 сек + (2048 * (32 * 64) пикс. * 1 бит/пикс.) / 1024 бит/сек = 6144 сек.*

*T_B = 2048 * log_2(128) бит / 1024 бит/сек + 2048 * 3 сек = 6158 сек.*

Следовательно, более эффективным является алгоритм А, а разница составит 14 секунд. Ответ А 14.

3. Основы логики (3 балла)

[Логический преобразователь]

Знайка построил логический преобразователь, который работает по следующему алгоритму:

1. На вход преобразователя подается восьмиразрядное восьмеричное число N_8.
2. Каждая цифра числа N_8 переводится в двоичную систему счисления и представляется тремя двоичными разрядами A_i, B_i, C_i, $1 \le i \le 8$. Старшему разряду двоичной записи i-той цифры N_8 соответствует A_i, а младшему – C_i.
3. Каждая такая тройка двоичных разрядов используется в качестве аргументов некоторой логической функции F(A,B,C). Значение 1 трактуется как «истина», а значение 0 – как «ложь». В результате вычисления функции $F(A_i,B_i,C_i)$ полученный результат записывается как i-тый разряд восьмиразрядного двоичного числа M_2. И в числе N_8 и в числе M_2 i=1 соответствует старшему разряду, а i=8 – младшему.
4. Число M_2 переводится в десятичную систему счисления и на выходе получается число M_{10}.

Определите, какая логическая функция F(A,B,C) реализована в преобразователе, если известно, что число **N_8=37421650** он преобразовал в **M_{10}=107**.

В ответе укажите формулу, которая может содержать логические переменные A, B и C и не более чем три логические операции. В качестве логических операций могут использоваться только операции отрицания, конъюнкции или дизъюнкции.

*Комментарий по вводу ответа: операнды вводятся большими **латинскими** буквами; логические операции обозначаются, соответственно как **not**, **and** и **or**. Запись не должна содержать скобок.*

Пример записи ответа: A or not B

Ответ: A and C or not B || C and A or not B || not B or A and C || not B or C and A

Решение

Число N_8 содержит ровно восемь неповторяющихся цифр от 0 до 7 включительно, каждая из которых может быть представлена трехразрядным двоичным числом, по условию задачи, интерпретируемым как аргументы логической функции от трех переменных, где значение 1 трактуется как «истина», а значение 0 – как «ложь». Также нам известны значения функции для каждого набора аргументов – их легко получить, переведя число 107 в двоичную систему счисления (дополним результат перевода до восьми разрядов незначащим нулем и получим 01101011), и интерпретировав разряды как истина и ложь так же как мы сделали это выше. Таким образом, мы имеем 8 уникальных комбинаций из значений истина и ложь и значения логической функции для каждой из этих комбинаций, то есть можно говорить, что нам задана полная таблица истинности логической функции от 3-х переменных (слева добавлен столбец со значениями разрядов восьмеричного числа N):

	A_i	B_i	C_i	M_i
3	0	1	1	0
7	1	1	1	1
4	1	0	0	1
2	0	1	0	0
1	0	0	1	1
6	1	1	0	0
5	1	0	1	1
0	0	0	0	1

Упорядочив таблицу, получаем:

A	B	C	F(A,B,C)
0	0	0	1

0	*0*	*1*	*1*
0	*1*	*0*	*0*
0	*1*	*1*	*0*
1	*0*	*0*	*1*
1	*0*	*1*	*1*
1	*1*	*0*	*0*
1	*1*	*1*	*1*

Теперь можно восстановить функцию по таблице истинности, например, построив СКНФ. Можно также заметить, что функция имеет истинное значение во всех случаях, когда аргумент B является истинным, независимо от значений аргументов A и B и кроме того имеет истинное значение в одном случае, когда аргумент B является ложным, но аргументы A и C являются истинными. Следовательно, это функция not B or A and C.

4. Алгоритмизация и программирование. Формальные исполнители (3 балла) [Шифратор]

Одним из методов шифрования текстов является перестановка символов в исходном тексте по определенному алгоритму. Цветик предложил для шифрования строки текста из N символов заданное количество раз выполнить следующую последовательность операций:

1. Для *очередного символа* строки найти относительно него третий символ справа, считая строку замкнутой в кольцо. То есть, если позиция искомого символа оказывается больше, чем номер последнего символа в строке, отсчет продолжается с начала строки. Например, относительно предпоследнего символа строки искомым символом будет второй символ строки.
2. Поменять местами эти два символа.
3. Перейти к следующему символу в строке, считая строку замкнутой в кольцо. То есть, если на текущем шаге *очередным символом* являлся последний символ строки, то на следующем шаге *очередным символом* будет первый символ строки.

На первом шаге очередным символом считается первый символ строки.

Пример: если взять строку “**intel**” и шесть раз выполнить указанную последовательность операций, то получится строка “**linte**”.

Цветик зашифровал некоторую исходную строку, выполнив указанную последовательность операций 200 раз, и получил в результате строку “**pcoineperm**”. Какую исходную строку зашифровал Цветик? В ответе укажите эту строку.

Ответ: coppermine

Решение

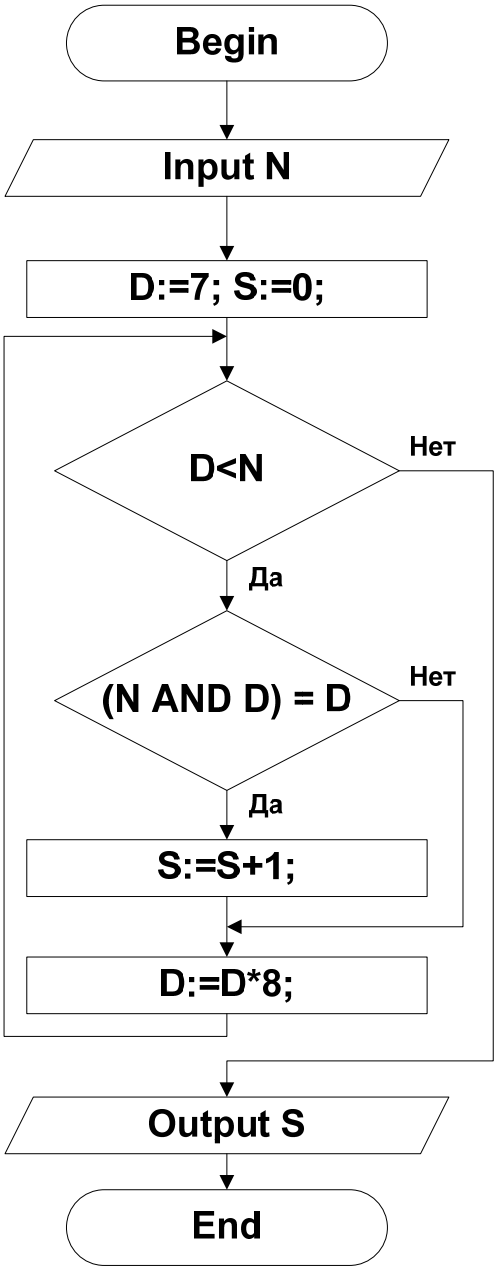
Обратим внимание, что результирующая строка состоит из 10 элементов, а описанный алгоритм может только изменять взаиморасположение символов строки на каждом шаге. Формализуем описанный алгоритм как переупорядочивание массива из 10 элементов и реализуем его в программном коде, сформировав перед началом его выполнения массив вида ‘0123456789’. Например, на языке Pascal алгоритм может быть реализован следующим образом:

```
var
mas : array [1..10] of integer;
i,first,second,t : integer;
begin
for i:=0 to 9 do mas[i+1]:=i;
for i:=1 to 200 do
   begin
   first:=(i-1) mod 10 + 1;
   second:=(i+2) mod 10 + 1;
   t:=mas[second];
   mas[second]:=mas[first];
   mas[first]:=t;
   end;
for i:=1 to 10 do write(mas[i]);
end.
```

В результате выполнения программы будет выведен массив 2017893456. Это позволяет нам определить позиции, на которых оказались элементы исходного массива после переупорядочивания. Значит, первый символ исходной строки соответствует второму символу результирующей, второй символ исходной строки – третьему символу результирующей, третий символ исходной строки – первому символу результирующей и т.д. Это позволяет нам восстановить исходную строку: coppermine.

5. Алгоритмизация и программирование. Анализ блок-схем (3 балла) [Большие числа]

Дана блок-схема алгоритма. Сколько существует различных целых положительных чисел **N**, меньших 2^{33} таких, что если их подавать на вход алгоритма, то на выходе будет получаться значение переменной **S=10**? Операция **N AND D** осуществляет побитовое логическое умножение переменных **N** и **D**. В ответе укажите целое число.\



Ответ: 77

Решение

*Сразу отметим, что решение задачи «в лоб» путем реализации в коде предложенной блок-схемы и перебора входных значений нецелесообразно, поскольку потребует значительного времени выполнения программы, поэтому попробуем решить задачу аналитически. Обратим внимание, что переменная S будет увеличивать на единицу свое значение только в одном случае, когда результат побитовой операции N and D будет совпадать со значением D. Также заметим, что первоначально $D=111_2$, затем после умножения на 8 становится равно 111000_2, после следующего умножения 111000000_2 и т.д. Каждый раз происходит сдвиг предыдущего числа влево на три разряда с приписыванием справа трех нулей. То есть числа D на каждом шаге алгоритма, это тройка единиц, после которой следует 0 или более троек нулей. Результаты побитового умножения таких чисел D на N будут равны самим этим числам только в одном случае, когда разряды числа N, соответствующие единицам в числе D после очередного сдвига также будут содержать только единицы. Числа, меньшие 2^{33} могут содержать не более 33 двоичных разрядов. Их можно представить как 11 последовательностей из трех двоичных разрядов. Для того, чтобы на выходе алгоритма получалось S=10, 10 из 11 троек разрядов должны содержать только двоичные единицы, а одиннадцатая тройка должна содержать хотя бы один 0 (если все 33 разряда будут единицами, значение S на выходе будет равно 11). Таким образом, тройка разрядов, не дающая выполнения условия N and D = D, может располагаться на одной из 11 возможных позиций и при этом такая тройка должна содержать не менее одного нуля (7 вариантов). Следовательно, искомых чисел будет 11*7=77.*

6. Алгоритмизация и программирование. Анализ кода (1 балл)

[Диагонали]

Дан фрагмент программы, вычисляющей значение целочисленной переменной S, исходя из значений двумерного целочисленного массива ***A***, размером 7 на 7 элементов:

$$A = \begin{bmatrix} 1 & 2 & 3 & 1 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 & 3 & 1 & 2 \\ 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 & 3 & 1 & 2 \\ 3 & 1 & 2 & 3 & 1 & 2 & 3 \\ 1 & 2 & 3 & 1 & 2 & 3 & 1 \end{bmatrix}$$

Бейсик	Паскаль	Алгоритмический

S=0
K=1
FOR I=1 TO 7
  FOR J=1 TO 7
  IF (K MOD X) = 0 THEN
  S=S+A(J,I)
  END IF
  K=K+1
  NEXT J
NEXT I
	S:=0;
	k:=1;
	for i:=1 to 7 do
	  for j:=1 to 7 do
	  begin
	  if (k mod X) = 0 then
	  S:=S+A[j,i];
	  k:=k+1;
	  end;
		S:=0
		k:=1
		нц для i от 1 до 7
		  нц для j от 1 до 7
		  если mod(k,X)=0
		  то S:=S+A[j,i]
		  все
		  k:=k+1
		  кц
		кц

Какое **минимальное** положительное значение целочисленной переменной **X** должно быть перед началом выполнения этого фрагмента, чтобы после его выполнения получилось значение переменной S=6?

В ответе укажите целое число.

При обращении к элементам массива, первый индекс обозначает номер строки, а второй индекс – номер столбца. Индексация элементов массива начинается с 1.

Ответ: 13

Решение

Разберем приведенный код. Алгоритм предполагает работу с переменной k, которая соответствует порядковому номеру элемента массива, считая от верхнего левого угла к правому нижнему углу. При этом при нулевом остатке от деления порядкового номера элемента на значение переменной X этот элемент суммируется с подобными элементами в переменной S.

По условию задачи переменная S=6 по окончании работы алгоритма, то есть после обработки ВСЕГО массива A. Значит, мы можем сложить два элемента со значение 3, три элемента со значением 2 и т.д.

Пронумеруем все элементы массива:

1/1	*2/2*	*3/3*	*1/4*	*2/5*	*3/6*	*1/7*
2/8	*3/9*	*1/10*	*2/11*	*3/12*	*1/13*	*2/14*
3/15	*1/16*	*2/17*	*3/18*	*1/19*	*2/20*	*3/21*
1/22	*2/23*	*3/24*	*1/25*	*2/26*	*3/27*	*1/28*
2/29	*3/30*	*1/31*	*2/32*	*3/33*	*1/34*	*2/35*
3/36	*1/37*	*2/38*	*3/39*	*1/40*	*2/41*	*3/42*
1/43	*2/44*	*3/45*	*1/46*	*2/47*	*3/48*	*1/49*

Нас просят найти минимальное значение переменной X. Логично предположить, что при этом количество суммируемых элементов должна быть максимальной. Что позволяет предположить необходимость суммировать значения элементов массива = 1. Минимальные порядковые номера у элементов со значением 1: 1,4,7,10,13. Проверим наше предположение:

При значении переменной X кратной 1 мы просуммируем все элементы массива, что нам не подходит.

При значении переменной X кратной 4 мы просуммируем элементы 4,8,12 и т.д. Сумма первых трех элементов этого ряда уже дает нам значение 6.

Аналогично, при проверке, нам не подойдут значения переменной X кратные 7 и 10.

А вот значение 13, даст нам необходимое значение переменной S.

Остается проверить, что нет значений переменной X, удовлетворяющих условиям меньших найденного. При этом видно, что для всех значений переменной X меньших найденного, будет достаточно сложить не более 4 кратных элементов.

7. Телекоммуникационные технологии (2 балла)

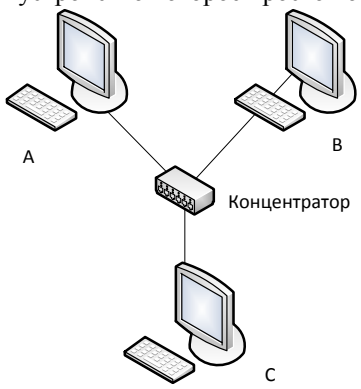
[Коллизии]

Авоська и Небоська, изучая коммуникационные технологии, разработали прототип компьютерной сети подобной Ethernet. В основе работы своей сети коротышки заложили следующие принципы:

1. Данные передаются сообщениями - кадрами с заголовком в 15 байт, признаком конца кадра в 5 байт и полем данных размером до 1480 байт.
2. Все события происходят с интервалами времени, кратными одной секунде.
3. Любой узел может начать передачу данных в любой момент времени, если сеть свободна.
4. Между передачей последовательных кадров одним и тем же узлом, этот узел делает технологическую паузу, чтобы другой узел мог начать передачу своего кадра.

5. Если узлу требуется осуществить передачу, а сеть занята, то он ожидает ее освобождения и **сразу** начинает передачу. Узлы знают, что сеть занята, если в предыдущую секунду один из узлов осуществлял передачу кадра и не закончил ее. Узлы знают, что сеть свободна, если в предыдущую секунду один из узлов закончил передачу кадра или в предыдущую секунду не осуществлялось передачи данных.
6. После того, как узел **целиком** получил кадр, он сравнивает поле «аппаратный адрес получателя» в заголовке кадра с адресом своего принявшего кадр адаптера, и в случае совпадения принимает кадр для обработки. В противном случае кадр на узле удаляется.
7. Если два или более узлов начинают передачу одновременно, то их сигналы смешиваются, и распознать передаваемые ими данные невозможно (эта ситуация называется коллизией).
8. Узлы, вступившие в коллизию, по обнаружении коллизии обрывают передачу и ожидают до повторной попытки передачи период, определяемый как случайное число из заданного диапазона различного для каждого узла. Однако между фактическим началом коллизии и ее обнаружением на конфликтующих узлах проходит некоторое время. Отсчет периода задержки начинается после истечения этого времени.

Авоська и Небоська построили сеть по топологии звезда (см. схему). В центре звезды они установили концентратор (hub или хаб) – устройство которое просто копирует сигнал на все свои порты кроме того, на который пришел сигнал.



В первой версии прототипа друзьям не удалось реализовать расчет случайной задержки (п.8) и они задали для каждого узла фиксированное значение задержки.

Если один узел должен передать несколько наборов данных, передача идет по порядку их постановки в очередь.

Каждый узел в сети в один момент времени или только передает, или только принимает сигнал (т.е. узлы работают в полудуплексном режиме).

При расчетах следует пренебречь конечной скоростью распространения сигнала и задержками на сетевых интерфейсах и промежуточных устройствах.

Условия задачи:

1. Узел **A** должен передать сначала узлу **B** 5920 байт данных, а затем узлу **C** 2960 байт данных.
2. Узел **B** должен передать узлу **C** 4440 байт данных.
3. Узел **C** должен передать узлу **B** 8880 байт данных.
4. Технологическая пауза (п. 4) = 3 сек.
5. Время необходимое для обнаружения коллизии всеми участниками = 2 сек.
6. Скорость передачи 2000 бит/сек.
7. Узел **A** начинает передачу первым. По истечении 16 секунд после этого начинает передавать узел **B**, а по истечении 44 секунд после начала передачи узлом **A** начинает передавать узел **C**.
8. Значения задержек в секундах для узла **A** = 3, для узла **B** = 4, а для узла **C** = 2.

Определите, через сколько секунд от начала передачи узлом **A** данных завершится передача указанных в условии данных всеми узлами. В ответе укажите целое число секунд.

Ответ: 107

Решение

Проанализируем описанную модель. Размер кадра составляет 1480 + 15 + 5 = 1500 байт = 12000 бит. Следовательно, при скорости передачи 2000 бит/сек, время передачи одного кадра будет составлять 6 секунд. Определим, какое количество кадров понадобиться для передачи обозначенных в тексте условия данных:

5920 байт = 5920/1480 = 4 кадра
2960 байт = 2960/1480 = 2 кадра
4440 байт = 4440/1480 = 3 кадра
8880 байт = 8880/1480 = 6 кадров

Теперь рассмотрим посекундно состояние телекоммуникационной системы:

Номер секунды	*Состояние*
1	*Начало передачи первого кадра от узла A к узлу B*
2	*Передача первого кадра от узла A к узлу B*
3	*Передача первого кадра от узла A к узлу B*
4	*Передача первого кадра от узла A к узлу B*

5	*Передача первого кадра от узла A к узлу B*
6	*Окончание передачи первого кадра от узла A к узлу B*
7	*Технологическая пауза*
8	*Технологическая пауза*
9	*Технологическая пауза*
10	*Начало передачи второго кадра от узла A к узлу B*
11	*Передача второго кадра от узла A к узлу B*
12	*Передача второго кадра от узла A к узлу B*
13	*Передача второго кадра от узла A к узлу B*
14	*Передача второго кадра от узла A к узлу B*
15	*Окончание передачи второго кадра от узла A к узлу B*
16	*Технологическая пауза*
17	*Начало передачи первого кадра от узла B к узлу C*
18	*Передача первого кадра от узла B к узлу C*
19	*Передача первого кадра от узла B к узлу C*
20	*Передача первого кадра от узла B к узлу C*
21	*Передача первого кадра от узла B к узлу C*
22	*Окончание пердачи первого кадра от узла B к узлу C*
23	*Начало передачи третьего кадра от узла A к узлу B*
24	*Передача третьего кадра от узла A к узлу B*
25	*Передача третьего кадра от узла A к узлу B*
26	*Передача третьего кадра от узла A к узлу B*
27	*Передача третьего кадра от узла A к узлу B*
28	*Окончание передачи третьего кадра от узла A к узлу B*
29	*Начало передачи второго кадра от узла B к узлу C*
30	*Передача второго кадра от узла B к узлу C*
31	*Передача второго кадра от узла B к узлу C*
32	*Передача второго кадра от узла B к узлу C*
33	*Передача второго кадра от узла B к узлу C*
34	*Окончание пердачи второго кадра от узла B к узлу C*
35	*Начало передачи последнего кадра от узла A к узлу B*
36	*Передача последнего кадра от узла A к узлу B*
37	*Передача последнего кадра от узла A к узлу B*
38	*Передача последнего кадра от узла A к узлу B*
39	*Передача последнего кадра от узла A к узлу B*
40	*Окончание передачи последнего кадра от узла A к узлу B*
41	*Начало передачи последнего кадра от узла B к узлу C*
42	*Передача последнего кадра от узла B к узлу C*
43	*Передача последнего кадра от узла B к узлу C*
44	*Передача последнего кадра от узла B к узлу C*
45	*Передача последнего кадра от узла B к узлу C, узел C ждет освобождения сети для начала передачи первого кадра узлу B*
46	*Окончание пердачи последнего кадра от узла B к узлу C*
47	*Начало передачи первого кадра от A к C, начало передачи первого кадра от C к B. КОЛЛИЗИЯ*
48	*Коллизия обнаружена*
49	*Узлы A и C ждут указанное время до попытки повторной передачи кадров*
50	*Узлы A и C ждут указанное время до попытки повторной передачи кадров. Для узла C время задержки закончилось.*
51	*Начало передачи первого кадра от узла C к узлу B*

52	*Передача первого кадра от узла С к узлу В*
53	*Передача первого кадра от узла С к узлу В*
54	*Передача первого кадра от узла С к узлу В*
55	*Передача первого кадра от узла С к узлу В*
56	*Окончание передачи первого кадра от узла С к узлу В*
57	*Начало передачи первого кадра от узла А к узлу С*
58	*Передача первого кадра от узла А к узлу С*
59	*Передача первого кадра от узла А к узлу С*
60	*Передача первого кадра от узла А к узлу С*
61	*Передача первого кадра от узла А к узлу С*
62	*Окончание передачи первого кадра от узла А к узлу С*
63	*Начало передачи второго кадра от узла С к узлу В*
64	*Передача второго кадра от узла С к узлу В*
65	*Передача второго кадра от узла С к узлу В*
66	*Передача второго кадра от узла С к узлу В*
67	*Передача второго кадра от узла С к узлу В*
68	*Окончание передачи второго кадра от узла С к узлу В*
69	*Начало передачи последнего кадра от узла А к узлу С*
70	*Передача последнего кадра от узла А к узлу С*
71	*Передача последнего кадра от узла А к узлу С*
72	*Передача последнего кадра от узла А к узлу С*
73	*Передача последнего кадра от узла А к узлу С*
74	*Окончание передачи последнего кадра от узла А к узлу С*
75	*Начало передачи третьего кадра от узла С к узлу В*
76	*Передача третьего кадра от узла С к узлу В*
77	*Передача третьего кадра от узла С к узлу В*
78	*Передача третьего кадра от узла С к узлу В*
79	*Передача третьего кадра от узла С к узлу В*
80	*Окончание передачи третьего кадра от узла С к узлу В*
81	*Технологическая пауза*
82	*Технологическая пауза*
83	*Технологическая пауза*
84	*Начало передачи четвертого кадра от узла С к узлу В*
85	*Передача четвертого кадра от узла С к узлу В*
86	*Передача четвертого кадра от узла С к узлу В*
87	*Передача четвертого кадра от узла С к узлу В*
88	*Передача четвертого кадра от узла С к узлу В*
89	*Окончание передачи четвертого кадра от узла С к узлу В*
90	*Технологическая пауза*
91	*Технологическая пауза*
92	*Технологическая пауза*
93	*Начало передачи пятого кадра от узла С к узлу В*
94	*Передача пятого кадра от узла С к узлу В*
95	*Передача пятого кадра от узла С к узлу В*
96	*Передача пятого кадра от узла С к узлу В*
97	*Передача пятого кадра от узла С к узлу В*
98	*Окончание передачи пятого кадра от узла С к узлу В*
99	*Технологическая пауза*
100	*Технологическая пауза*

101	*Технологическая пауза*
102	*Начало передачи последнего кадра от узла C к узлу B*
103	*Передача последнего кадра от узла C к узлу B*
104	*Передача последнего кадра от узла C к узлу B*
105	*Передача последнего кадра от узла C к узлу B*
106	*Передача последнего кадра от узла C к узлу B*
107	*Окончание передачи последнего кадра от узла C к узлу B. ЗАКОНЧЕНА ПЕРЕДАЧА ВСЕХ ДАННЫХ*

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

[Главное правильно взвесить]

Шесть коротышек участвовали в соревновании. Соревнование состоит из четырех испытаний: A, B, C и D. В каждом из испытаний участник мог набрать от 0 до 10 баллов. Результаты участников были занесены в электронную таблицу:

	A	B	C	D	E	F
1		A	B	C	D	Итог
2	Торопыжка	10	4	6	2	
3	Авоська	4	7	5	7	
4	Небоська	3	6	8	4	
5	Пончик	2	3	7	9	
6	Незнайка	5	6	8	2	
7	Ворчун	4	9	3	5	
8						
9						
10						

Для определения победителя жюри рассчитывает для каждого участника его итоговый результат соревнований. Для этого в ячейку **F2** помещают следующую формулу:

=B$9*$B2+C$9*$C2+D$9*$D2+E$9*$E2

Затем ячейку **F2** копируют во все ячейки диапазона **F3:F7**.

В ячейках **B9**, **C9**, **D9** и **E9** находятся весовые коэффициенты, зависящие от сложности отдельного испытания, и определяющие вклад баллов, набранных за испытание, в итоговую оценку.

Известно, что в качестве весовых коэффициентов использовались только числа: 0,7; 0,9; 1,1; 1,3, причем у всех испытаний они различны, но не известно, для какого из испытаний какой весовой коэффициент был назначен.

Определите, какие весовые коэффициенты были назначены для какого из испытаний, если после расчета итоговых результатов максимальный результат оказался у Незнайки. В ответе укажите через пробел 4 числа: сначала значение в ячейке **B9**, затем значение в ячейке **C9**, затем значение в ячейке **D9** и, наконец, значение в ячейке **E9**.

Ответ: 0,9 1,1 1,3 0,7 || 0.9 1.1 1.3 0.7

9. Базы данных (1 балл)

[Конвейер]

Завод занимается сборкой изделий из деталей. Для получения каждого экземпляра изделия требуется определенное количество деталей нескольких видов. Некоторые детали также можно собирать из других деталей. Часть деталей уже собрана или закуплена и хранится на складе, а часть можно собрать из имеющихся деталей, чтобы потом использовать для сборки конечных изделий. База данных автоматизированной системы управления технологическим процессом включает в себя в частности две таблицы: «Состав деталей и изделий» и «Склад».

В таблице «Склад» хранятся данные о текущем количестве деталей каждого вида. Конечному изделию соответствует идентификатор 9.

Склад		
ID_детали	**Название_детали**	**Количество**
1	Деталь_1	20
2	Деталь_2	20
3	Деталь_3	25
4	Деталь_4	40
5	Деталь_5	15
6	Деталь_6	20
7	Деталь_7	25
8	Деталь_8	35
9	Изделие	0

В таблице «Состав деталей и изделий» хранятся данные о том, какое количество каких деталей требуется для сборки другой детали или изделия в целом. Значения идентификаторов в столбцах «ID_собираемой_детали» и «ID_используемой_детали» берутся из таблицы «Склад». Наличие записи в таблице «Состав деталей и изделий» говорит о том, что для сборки детали или изделия с идентификатором «ID_собираемой_детали» необходимы детали с

идентификатором «ID_используемой_детали» в количестве, указанном в столбце «Количество». Деталь или изделие могут быть изготовлены только, если в наличии есть достаточное количество всех необходимых для их изготовления деталей. Одна и та же деталь может быть необходима для изготовления нескольких других деталей.

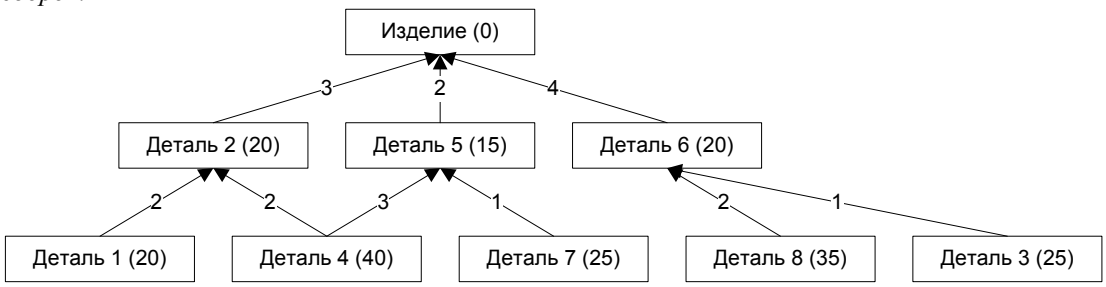
Состав деталей и изделий		
ID_собираемой_детали	**ID_используемой_детали**	**Количество**
9	2	3
9	5	2
9	6	4
2	1	2
2	4	2
5	4	3
5	7	1
6	8	2
6	3	1

Определите, какое максимальное количество изделий можно собрать, если исходное количество деталей соответствует записям в приведенной выше таблице «Склад». В ответе укажите целое число.

Ответ: 9

Решение

На основе данных таблиц построим дерево конструирования деталей. На стрелках подписано количество деталей, необходимых для производства детали следующего уровня. В скобках указано исходное количество деталей до начала сборок.



Заметим, что только Деталь 4 нужна для изготовления сразу двух других деталей – 2 и 5. Проанализировав варианты разделения исходного количества деталей 4 для изготовления деталей 2 и 5, увидим, что оптимальным будет соотношение: 14 деталей 4 для изготовления деталей 2 и 26 деталей 4 для изготовления деталей 5. Тогда выполнив сборку дополнительного количества деталей 2, 5 и 6 получим 20+7=27 деталей 2, 15+8=23 детали 5 и 20+17=37 деталей 6. Теперь соберем из них изделия. Деталей 2 хватит на 9 изделий, деталей 5 – на 12 изделий, а деталей 6 – на 9 изделий. Следовательно максимальное количество изделий, которое можно собрать по этой схеме равно 9.

10. Технологии программирования (2 балла) [Автоматическая погрузка]

На складе вашей компании теперь работают новые роботы-грузчики. Работают они следующим образом: на склад загоняются две пустые фуры, которые для удобства занумерованы **1** и **2**. Роботы, перед тем как погрузить очередной груз, считают суммарную массу груза в каждой фуре, и кладут очередной груз в фуру, которая на текущий момент имеет наименьший суммарный груз. Если же суммарные грузы равны, то груз кладут в фуру под номером **2**.

Начальник ИТ отдела не доверяет роботам, поэтому поручает вам разработать программу, которая, зная всю информацию о грузах, выведет предполагаемую последовательность действий роботов.

Формат входного файла

В первой строке входного файла **input.txt** находится натуральное число **n** (**1 ≤ n ≤ 100**) — количество грузов. Во второй строке содержится **n** натуральных чисел **x** (**1 ≤ x ≤ 100**) — массы грузов в порядке погрузки.

Формат выходного файла

В выходной файл **output.txt** требуется вывести **n** чисел, разделенных пробелами — последовательность из **1** и **2**, **i**-й член которой является номером фуры, в которую будет погружен груз с номером **i**.

Пример входных и выходных данных

input.txt	output.txt
4 5 3 2 2	2 1 1 2
6 10 1 1 1 10 7	2 1 1 1 1 2

11. Технологии программирования (4 балла) [Упаковка коробок]

Вы разрабатываете программное обеспечение для организации, занимающейся упаковкой коробок. Ваша цель — по заданным параметрам коробок определить, есть ли возможность упаковать все коробки в одну или нет.

Процесс упаковки коробок происходит по следующему правилу:

- одну коробку можно упаковать в другую, если длина, высота и ширина первой строго меньше, соответственно, длины, высоты и ширины второй

- в коробку можно поместить не более одной коробки, но, возможно, внутри помещаемой коробки есть другие коробки
- коробки можно поворачивать относительно горизонтальной оси (длина коробки при этом остается неизменной, а высота и ширина меняются местами)

Вам дано описание размеров всех коробок. Если возможно упаковать коробки по заданным правилам, выведите номера коробок в получившейся конструкции, начиная с номера внешней коробки. В противном случае выведите **-1**.

Формат входного файла

В первой строке входного файла **input.txt** находится натуральное число **n**(**2 ≤ n ≤ 1000**) — количество коробок. В следующих **n** строках содержится по три по три целых числа **l**, **w** и **h** (**1 ≤ l, w, h ≤ 100000**) — длина, ширина и высота очередной коробки соответственно. Коробки пронумерованы начиная с 1.

Формат выходного файла

В выходной файл **output.txt** выведите **n** различных чисел от 1 до **n** в случае, если ответ существует, и -1 в противном случае.

Пример входных и выходных данных

input.txt	output.txt
5 5 4 6 7 10 5 4 3 5 1 1 1 2 4 2	2 1 3 5 4