Распределение регистров

Современные компьютерные процессоры имеют ограниченное число *регистров*. Это места хранения общего назначения, которые работают существенно быстрее главной памяти. Операции вычисляющие функции (сложение, умножение и т.д.) берут свои аргументы из регистров и результат помещают также в регистр.

В этом упражнении мы рассмотрим задачу *распределения регистров* для расчёта значения выражения. Внутри компьютера выражение представляется в виде дерева. Листья (конечные вершины) этого дерева соответствуют значениям, которые должны быть загружены из главной памяти. Промежуточные вершины соответствуют операциям, и каждая из них имеет столько потомков (поддеревьев), сколько аргументов у операции. Очевидно, что значения всех аргументов операции должны быть известны к тому моменту, когда операция выполняется.

Так как имеется лишь ограниченное число регистров, компилятор должен решить, которые из промежуточных результатов следует хранить в регистрах (они доступны всегда, когда они нужны) и какие держать в главной памяти (их придётся загружать перед тем, как использовать). Также может оказаться полезным менять порядок вычисления аргументов операции (вот почему большинство языков высокого уровня не гарантируют какого-либо конкретного порядка).

Ваша задача — написать программу, которая для заданного дерева выражения составляет план распределения регистров и порядок вычисления, требующие наименьших затрат.

Входные данные. В первой строке текстового файла `REGS.IN` дано число регистров N ($1 \leq N \leq 100$). Вторая строка содержит два целых числа: затраты на загрузку значения из главной памяти в регистр C_l ($1 \leq C_l \leq 100$) и затраты на сохранение значения из регистра в главную память C_s ($1 \leq C_s \leq 100$). Оставшаяся часть файла описывает дерево выражения, начиная с корневой вершины:

- первая строка содержит число потомков вершины, K_x ($0 \leq K_x \leq 10$ и $K_x \leq N$);
- если $K_x = 0$, это конечная вершина (лист), и её описание на этом завершено;
- если $K_x > 0$, это промежуточная вершина, и:
  - следующая строка содержит одно целое число: затраты C_x ($1 \leq C_x \leq 100$) на операцию, которой соответствует данная вершина;
  - и далее следуют описания K_x поддеревьев по такой же схеме.

Вершины пронумерованы от 1 до M в порядке их появления во входном файле. Можно предполагать, что $M \leq 10\,000$.

Выходные данные. Первая строка выходного файла `REGS.OUT` должна содержать минимальную сумму затрат на расчёт значения выражения. Оставшаяся часть файла должна содержать по одной строке на каждую промежуточную вершину дерева. Каждая строка должна содержать два целых числа: первое – номер вершины, значение которой необходимо подсчитать, а второе равно 1, если результат следует оставить в регистре, и 0, если следует выгрузить в главную память (для чего потребуется прибавить C_s к общей сумме затрат).

Операции должны быть перечислены в том порядке, в котором их следует выполнять, чтобы сумма затрат на вычисление минимально возможная, согласно следующим условиям:

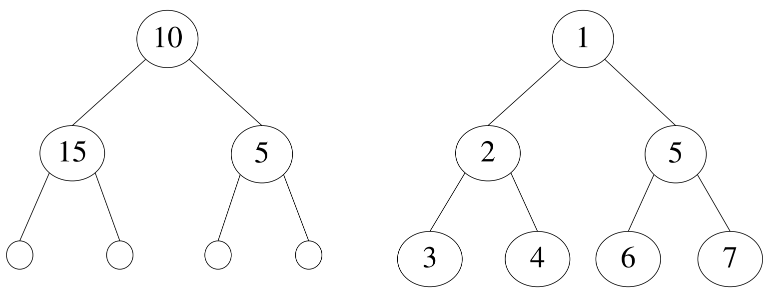
- Значение вершины может быть вычислено только после того, как вычислены значения всех её потомков;
- все аргументы выполняемой операции, значения которых находятся не в регистре, должны быть предварительно загружены из главной памяти (сответственно с затратами C_l на каждое значение);
- регистры, содержащие значения аргументов выполненной операции, могут быть использованы заново немедленно, даже для сохранения результата операции.

Если существует несколько планов с минимальной суммой затрат, вывести любой из них.

Пример.

REGS.IN	REGS.OUT
2	47
3 2	2 0
2	5 1
10	1 1
2	
15	
0	
0	
2	
5	
0	
0	

Замечание. Вышеприведённый входной файл иллюстрирует следующий рисунок: оба дерева соответствуют дереву выражения, причём дерево слева показывает стоимость промежуточных вершин, а дерево справа показывает нумерацию вершин.



Сумма затрат плана в выходном файле равна

$$(C_l + C_l + 15 + C_s) + (C_l + C_l + 5) + (C_l + 10) = 47.$$

Замечание. Вам будет дана программа `REGSCHECK`, которая проверяет корректность (но не оптимальность) файла `REGS.OUT` относительно `REGS.IN` и выдаёт информацию об ошибках.