Учебно-тренировочные сборы команды Донецкой области на Всеукраинскую олимпиаду по информатике

IV тур, *3 марта 2013 г.*

1. **Футбол.** Олег – большой любитель футбола и статистики. Недавно он нашел результаты участия его любимой команды в каком-то давнем чемпионате. К сожалению, единственной сохранившейся информацией оказалось то, сколько матчей было сыграно и сколько очков набрала команда. Напоминаем, что если матч завершается победой команды, то ей присуждается три очка, ничьей – одно очко, и если команда проигрывает матч, то она не получает ни одного очка. Олегу стало интересно, сколько различных вариантов прохождения чемпионата было у его любимой команды. Различными считаются варианты, если результат хотя бы одного матча различен, причем счет не принимается во внимание, а учитывается только то, завершился матч победой, ничьей или проигрышем.

   Задание. Напишите программу, которая находит количество вариантов прохождения чемпионата командой.

   Входные данные. В единственной строке входного файла *football.dat* содержит два целых числа n и k ($0 \le n \le 120$, $1 \le k \le 40$), обозначающих количество набранных командой очков и количество матчей, сыгранных этой командой в чемпионате, соответственно.

   Выходные данные. В единственную строку выходного файла *football.sol* выведите число различных вариантов прохождения чемпионата любимой командой Олега.

   Примеры входных и выходных данных

football.dat	*football.sol*
3 2	2
4 3	6

2. **Горки.** *Горкой* будем называть неубывающую последовательность из двух и более чисел. *Высотой горки* назовем самый большой элемент в ней. *Гористым разбиением последовательности* назовем набор из минимального количества горок, таких, что если их записать поочередно слева направо, получим исходную последовательность.

   Задание. Напишите программу, которая расположит заданный набор чисел в такой последовательности, чтобы сумма высот горок в ее гористом разбиении будет максимальной.

   Входные данные. Первая строка входного файла *mountain.dat* содержит одно целое число N – количество чисел в наборе ($2 \le N \le 10^5$). Во второй строке записаны N целых чисел в пределах от 1 до 10^5.

   Выходные данные. В единственную строку выходного файла *mountain.sol* выведите одно число – максимально возможную сумму высот горок в последовательности.

   Примеры входных и выходных данных

mountain.dat	*mountain.sol*
4 1 2 3 4	7

   Замечание. Возможный вариант последовательности: 2 3 1 4. Ее гористое разбиение: (2 3) + (1 4). Высота первой горки 3, второй – 4. Сумма равна 7.

3. **Мэр.** Максим устал от того, что в его родном городе на тротуарах полно мусора, из года в год растет преступность в то время, как милиция бездействует, да еще ко всему прочему на улице идет дождь. Поэтому он решил баллотироваться на пост мэра, чтобы все изменить в лучшую сторону. Однако быть мэром в родном городе Максима не так просто. Мэру нужно знать всех жителей. И если все

же есть житель x, незнакомый мэру, то мэр должен знать по крайней мере кого-нибудь, кто знает этого жителя x (другими словами, мэр должен иметь с жителем x общего знакомого). Максим не уверен, может ли он стать мэром или нет.

Задание. Напишите программу, которая определит, удовлетворяет ли кандидатура Максима указанным для мэра требованиям.

Входные данные. В первой строке входного файла *mayor.dat* содержится два целых числа N и K ($1 \le N \le 3 \cdot 10^5, 1 \le K \le 10^6$), обозначающие количество жителей города и количество пар знакомых людей. В каждой из последующих K строк задаются два числа в пределах от 1 до N, определяющие пару знакомых друг с другом людей. Максим является жителем с номером 1.

Выходные данные. В выходной файл *mayor.sol* выведите одно число 0, если Максим может стать мэром. В противном случае выведите номера тех жителей, из-за которых Максим не сможет стать мэром (то есть тех, кого не знает ни Максим, ни его знакомые). Выводить числа можно в любом порядке, но каждое ровно один раз.

Примеры входных и выходных данных

mayor.dat	*mayor.sol*
4 3 1 3 1 4 4 2	0
6 5 1 2 1 3 2 3 2 5 4 5	6 4

4. **Ящик с зеркалами.** Есть черный ящик, который представляет из себя прямоугольник размера $M \times N$ (M строк, N столбцов). В каждой его клетке может быть расположено зеркало, ориентированное диагонально от левого нижнего угла клетки к правому верхнему. Обе стороны такого зеркала отражают свет. На сторонах ящика, напротив каждой строки и столбца есть отверстия. В любое отверстие вы можете посветить лучом света, но лишь в направлении перпендикулярном стороне, на которой расположено это отверстие. Нетрудно понять, что в такой луч при отражении от зеркала, будет менять свое направление на 90 градусов. Когда же луч проходит через пустую клетку, его направление не изменяется. Все отверстия пронумерованы числами от 1 до $2(M+N)$ в порядке обхода ящика по периметру против часовой стрелки, начиная с самой верхней клетки левой стороны. Поскольку ящик черный, то расположение зеркал внутри него нам не видно, и единственный способ определить это расположение – посветить лучом в некоторые отверстия и посмотреть, откуда свет выходит.

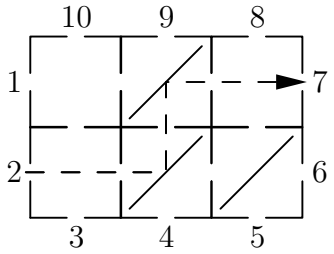
Задание. Напишите программу, которая определит возможное расположение зеркал внутри ящика.

Входные данные. В первой строке файла *mirror.dat* записаны два целых числа M и N ($1 \le M, N \le 100$), определяющие размеры ящика. Во второй строке записаны $2(M+N)$ целых чисел. i-ое число определяет номер отверстия из которого выйдет свет, если посветить лучом в отверстие с номером i. Гарантируется, что входные данные непротиворечивы, то есть существует по крайней мере одно расположение зеркал, удовлетворяющее условию.

Выходные данные. В выходной файл *mirror.sol* вывести M строк, в каждой из которых будет записано по N чисел, определяющих возможное расположение зеркал внутри ящика. j-ое число i-ой строки должно равняться 1, если есть зеркало в соответствующей клетке ящика, и 0, если эта клетка пуста. Если существует несколько возможных решений, выведите любое из них.

Примеры входных и выходных данных

mirror.dat	*mirror.sol*
2 3 9 7 10 8 6 5 2 4 1 3	0 1 0 0 1 1



5. **Интересная последовательность.** Последовательность чисел целых $a_1, a_2, \ldots$ определяется следующим образом: первый член ее член a_1 равен 0, а каждое последующее число a_i $(i > 1)$ определяется как наименьшее целое число, которое больше, чем a_{i-1}, а в его десятичной записи не содержится цифр, представленных в десятичной записи числа a_{i-1}.

   Задание. Напишите программу, которая которая по значению числа n вычисляет величину a_n.

   Входные данные. Единственная строка входного файла *numseq.dat* содержит одно целое число n $(1 \leq n \leq 500)$.

   Выходные данные. В единственную строку выходного файла *numseq.sol* выведите число a_n.

   Примеры входных и выходных данных

numseq.dat	*numseq.sol*
1	0
28	911