1. Последовательность

1 секунда 10 очков

Назовём переменами знака места в числовой последовательности, где за положительным элементом следует отрицательный или за отрицательным положительный так, что между ними либо вообще нет других елементов, либо там только нули.

Написать программу, которая найдёт количество перемен знака в заданной последовательности целых чисел.

Входные данные. На первой строке текстового файла `jada.sis` находится количество элементов последовательности N ($1 \leqslant N \leqslant 1000$). На второй строке файла находятся N разделённых пробелами целых чисел с абсолютными величинами до 10 000 — элементы последовательности.

Выходные данные. На единственную строку текстового файла `jada.val` вывести одно целое число — количество перемен знака в данной последовательности.

Пример.

jada.sis	jada.val
3 -1 1 -1	2

Пример.

jada.sis	jada.val
3 -1 0 1	1

2. Дата

1 секунда 20 очков

Рассмотрим представление дат в виде 8-значных чисел $AAAAKKPP$, где $AAAA$ обозначает (всегда четырёхзначный) год, KK (всегда двухзначный) месяц и PP (всегда двухзначный) номер дня.

Палиндромными назовём такие даты, представления которых одинаковы как при чтении слева направо, так и при чтении справа налево. Например, 2 ноября 2011 (20111102) — палиндромная дата, тогда как 2 декабря 2012 (20121202) нет.

Написать программу, которая найдёт ближайшую палиндромную дату, следующую за заданной датой.

Входные данные. На единственной строке текстового файла `kuup.sis` находится представление даты в виде $AAAAKKPP$.

Выходные данные. На единственную строку текстового файла `kuup.val` вывести представление в виде $AAAAKKPP$ самой ранней из следующих за данной датой палиндромных дат. Можно считать, что в каждом тесте такая дата (с не более чем четырёхзначным годом) найдётся.

При подсчёте високосных лет считать, что григорианский календарь используется с 1 по 9999 год. Это значит, что високосными следует считать те года, номер которых делится на 4, но не делится на 100, а также те, номер которых делится на 400.

Пример.

kuup.sis	kuup.val
20111101	20111102

Пример.

kuup.sis	kuup.val
20111102	20200202

3. Матрица

1 секунда 30 очков

Детерминантом матрицы (числовой таблицы) размером 3×3

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

называют выражение

$$(a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - (a_{31}a_{22}a_{13} + a_{32}a_{23}a_{11} + a_{33}a_{21}a_{12}).$$

Написать программу, которая найдёт сколько различных абсолютных величин детерминанта можно получить при расстановке данных 9 целых чисел в матрицу размером 3×3.

Входные данные. На единственной строке текстового файла `maat.sis` находятся 9 разделённых пробелами целых чисел, абсолютные величины которых не превышают 100.

Выходные данные. На единственную строку текстового файла `maat.val` вывести количество различных абсолютных величин детерминанта, полученных при всевозможных перестановках данных 9 чисел в матрице.

Пример.

maat.sis	maat.val
1 1 2 2 0 0 0 0 0	3

Возможные значения детерминанта равны 4 (например, матрица A_1 снизу), 2 (например, A_2), 0 (например, A_3), -2 (например, A_4) и -4 (например, A_5). Получить другие значения нельзя, поэтому различных абсолютных величин всего три: 0, 2 и 4.

A_1	A_2	A_3	A_4	A_5
$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 1 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 2 \\ 0 & 0 & 0 \\ 1 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 2 \\ 0 & 2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$

4. Выбери меня!

1 секунда 40 очков

В одном федеративном государстве выборы президента проводятся между двумя кандидатами в два тура. В первом туре голосуют все жители. У каждого округа есть определённое число представителей, которые во втором туре голосуют за кандидата, получившего в этом округе наибольшее число голосов в первом туре. Во втором туре побеждает кандидат, который получит наибольшее количество голосов представителей.

В обоих турах для победы нужно получить больше голосов чем соперник. Если в первом туре в каком-либо округе оба кандидата получают одинаковое число голосов, то представители этого округа не участвуют во втором туре. Ничья во втором туре означает повторные выборы.

Ситуация с нынешними выборами такова, что один из кандидатов решил подкупить избирателей, но этот план стал известен общественности. Как обычно, официальное расследование не смогло ничего доказать, и поэтому участие этого кандидата в выборах не запрещено напрямую. Однако теперь за этого кандидата голосуют только те, кому он платит; все остальные голосуют за его соперника.

Написать программу, которая найдёт минимальное множество избирателей, которых должен подкупить этот кандидат в первом туре, чтобы оказаться выбранным во втором.

Входные данные. На первой строке текстового файла `vali.sis` находится число округов N ($1 \leqslant N \leqslant 100$). На каждой из следующих N строк находятся два разделённых пробелом целых числа — число жителей M_i ($1 \leqslant M_i \leqslant 10^6$) и число представителей K_i ($1 \leqslant K_i \leqslant 10^3$, $K_i \leqslant M_i$) округа. Округи пронумерованы $1 \ldots N$ в порядке их появления во входных данных.

Выходные данные. На первую строку текстового файла `vali.val` вывести целое число K — число округов, жителей которых нужно подкупить. На каждую из следующих K строк вывести два разделённых пробелом целых числа — номер округа и число подкупаемых избирателей из этого округа. Если решений с минимальным числом подкупаемых избирателей несколько, вывести любое из них.

Пример.

`vali.sis`
```
3
121 2
150 3
140 4
```

`vali.val`
```
2
1 61
3 71
```

Пример.

`vali.sis`
```
3
120 2
150 3
140 4
```

`vali.val`
```
2
1 60
3 71
```

Оценивание. В тестах суммарной ценностью 20 очков $N < 15$.

5. Каменный город

1 секунда 50 очков

Мэр города желает поменять каменные плиты, покрывающие ратушную площадь. Из каменоломни можно заказать прямоугольные плиты любого размера. Мэр — перфекционист и поэтому желает покрыть площадь как можно меньшим числом плит.

Написать программу, которая прочитает данные об имеющихся прямоугольных плитах со сторонами направленными с севера на юг и с востока на запад, и найдёт минимальное число новых плит, которыми можно покрыть точно такую же территорию точно в один слой.

Входные данные. На первой строке текстового файла `kivi.sis` находится число имеющихся плит N ($1 \leqslant N \leqslant 500$). На каждой из следующих N строк находятся четыре разделённых пробелами целых числа — расстояния от северной и южной сторон плиты до северного края площади и от западной и восточной сторон до западного края. Все расстояния – целые числа $0 \ldots 10\,000$.

Можно считать, что площадь связна, т.е. из любой её точки возможно дойти до любой другой, не выходя за территорию площади. Также, любая прямая, направленная с севера на юг или с востока на запад, пересекает границы площади не более двух раз.

Выходные данные. На первую строку текстового файла `kivi.val` вывести минимальное возможное число новых плит K, а на каждую из следующих K строк вывести четыре разделённых пробелами целых числа — расстояния от северной и южной сторон плиты до северного края площади и от западной и восточной сторон до западного края. Если решений с минимальным числом плит несколько, вывести любое из них.

Пример.

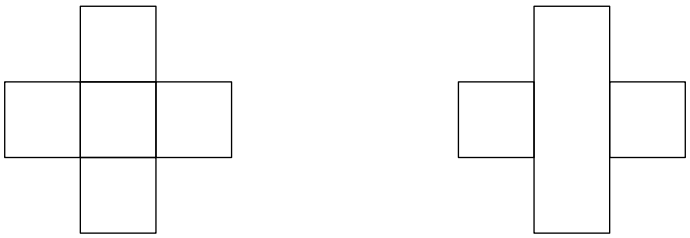
`kivi.sis`

```
5
20 30 10 20
10 20 20 30
20 30 20 30
30 40 20 30
20 30 30 40
```

`kivi.val`

```
3
20 30 10 20
10 40 20 30
20 30 30 40
```

На рисунке слева показано расположение старых плит, а справа — новых.



Оценивание. В тестах суммарной ценностью 20 очков $N < 15$.

6. Потребление электроэнергии

Тестирование 50 очков

Счётчик электроэнергии с возможностью удалённого чтения каждый час сообщает фирме-энергопоставщику среднюю мощность, потребляемую клиентом на протяжении этого часа. Энергопоставщик заказывает у IT-фирмы программу, которая должна на основании этих почасовых показаний подсчитать суммы дневного и ночного потребления электроэнергии клиентом за месяц.

В соответствии с условиями договора энергопоставщика дневным потреблением считается энергия, потребленная в период с 7:00 по 23:00 по рабочим дням в зимнее время, а также в период с 8:00 по 24:00 по рабочим дням в летнее время; всё остальное потребление считается ночным.

Летнее время начинается в последнее воскресенье марта. В этот день в 3:00 (по действующему на тот момент зимнему времени) часы переводятся на один час вперёд (то есть, часа 3:00–4:00 в этот день нет). Летнее время заканчивается в последнее воскресенье октября. В этот день в 4:00 (по действующему на тот момент летнему времени) часы переводятся на один час назад (то есть, час

3:00–4:00 в этот день будет два раза, один раз по летнему и один раз по зимнему времени).

Программа должна правильно учитывать и високосные года, то есть, в соответствии с григорианским календарём, те года, номер которых делится на 4, но не делится на 100, а также те, номер которых делится на 400.

Составить комлект тестовых данных для фирмы-энергопоставщика, с помощью которого она сможет проверить верна ли программа, написанная IT-фирмой, прежде, чем за неё заплатить.

Входные данные. На первой строке текстового файла `tarb.sis` находятся два разделённых пробелом целых числа A и K, где A — номер года, а K — номер месяца анализируемого периода ($1900 \leqslant A \leqslant 2100$, $1 \leqslant K \leqslant 12$). На второй строке файла находится количество показаний потребления N ($0 \leqslant N \leqslant 745$). На каждой из следующих N строк файла находятся два разделённых пробелом целых числа T и P, где T — порядковый номер часа показания в месяце и P — среднее потреблённой мощности во время этого часа ($0 \leqslant P \leqslant 1000$). Нумерация часов в каждом месяце начинается заново с единицы. Показания в файле находятся в хронологическом порядке, и показания о часах, во время которых клиент не потреблял электричества, могут отсутствовать.

Выходные данные. На единственной строке текстового файла `tarb.val` находятся два разделённых пробелом целых числа: суммы дневного и ночного потребления электроэнергии соответственно входным данным.

Пример.

`tarb.sis`

```
2012 1
5
1 2
2 3
26 3
32 6
744 0
```

`tarb.val`

```
6 8
```

Входные данные означают следующее:

Час	Мощность	Дата	День недели	Время	Вид
1	2 кВт	1 января	воскресенье	0:00–1:00	ночное
2	3 кВт	1 января	воскресенье	1:00–2:00	ночное
26	3 кВт	2 января	понедельник	1:00–2:00	ночное
32	6 кВт	2 января	понедельник	7:00–8:00	дневное
744	0 кВт	31 января	вторник	23:00–24:00	ночное

Таким образом дневное потребление равно 6 кВтч, а ночное равно $2 + 3 + 3 = 8$ кВтч.

Оценивание. В этом задании в качестве решения нужно предоставить один ZIP-файл содержащий до 15 входных файлов с именами `tarbtest.01.sis`, `tarbtest.02.sis`, ... и соответствующих им выходных файлов с именами `tarbtest.01.val`, `tarbtest.02.val`, ...

Если некая пара файлов не состоит из корректного входного файла и соответствующего ему выходного файла, то за эту пару очки не даются. Корректные пары используются при тестировании специальных программ с ошибками. Участник получит определённое число очков за каждую программу, которая выдаст неверный ответ хотя бы в одном его тесте.

7. Бинарные деревья

Анализ 50 очков

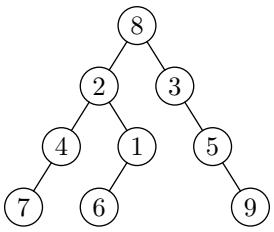
Пусть T бинарное дерево[1], все N вершин которого пронумерованы $1 \dots N$ (в каком-то неизвестном порядке). Рассмотрим схему, где при обработке вершины t выполняются следующие шаги[2]:

1. ...
2. Если у t есть левый сын v, обработать v рекурсивно по тому же алгоритму.
3. ...
4. Если у t есть правый сын p, обработать p рекурсивно по тому же алгоритму.
5. ...

Последовательность номеров вершин дерева T, которую получим, если в вышеприведённой схеме на шагу 1 выведём номер текущей вершины t (и на шагах 3 и 5 ничего не делаем) и начнём выполнение алгоритма с корня дерева, назовём прямым порядком $E(T)$.

Аналогично, внутренним порядком $K(T)$ назовём последовательность, которую получим, если выведём номер вершины t на шагу 3, и обратным порядком $L(T)$ — последовательность, которую получим, если выведём номер t на шагу 5.

Например у двоичного дерева



прямой порядок — 8, 2, 4, 7, 1, 6, 3, 5, 9, внутренний порядок — 7, 4, 2, 6, 1, 8, 3, 5, 9 и обратный порядок — 7, 4, 6, 1, 2, 9, 5, 3, 8.

Проанализировать, какие комплекты порядков бинарного дерева достаточны для однозначного восстановления структуры этого дерева. Это значит, нужно ответить на вопрос

Всегда ли достаточно знания комплекта X для однозначного восстановления дерева T?

для всех возможных комплектов порядков X (т.е. для $X = \langle E(T) \rangle$, ..., $X = \langle E(T); K(T) \rangle$, ..., $X = \langle E(T); K(T); L(T) \rangle$).

Если ответ утвердительный, описать алгоритм восстановления дерева. Алгоритм можно дать в виде словесного описания, но это описание должно быть достаточно точным для однозначного понимания алгоритма, например:

1. Пусть $E(T) = e_1, e_2, \dots, e_N$ и $L(T) = l_1, l_2, \dots, l_N$.
2. Из определения $E(T)$ знаем, что e_N ...
3. Поэтому можем найти такой l_i, что ...

...

Если ответ отрицательный, привести по крайней мере один контрпример. То есть привести два таких дерева T_1 и T_2, при которых все соответствующие порядки совпадают, но сами T_1 и T_2 различаются.

Оценивание. В этом задании в качестве решения нужно предоставить один текстовый файл, в котором обоснованы ответы для всех возможных комплектов порядков.

При оценивании очки снимаются за неразобранные случаи, неправильные ответы, а также при неполных обоснованиях (при утвердительном ответе неполные или неверные алгоритмы, при отрицательном ответе неполные или неверные контрпримеры).

[1] См. также `http://www.ut.ee/~at/prog/prog08.pdf`, абзац 8.1.2.
[2] См. также `http://www.ut.ee/~at/prog/prog08.pdf`, абзац 8.2.2.