Проверка решений

Проверка решений будет осуществляться при помощи тестирующей системы Ejudge (http://ejudge.ru) на заранее подготовленном наборе тестов. Решение засчитывается только в том случае, если оно проходит все тесты. Если возникают ошибки, Вы можете изучить протокол тестирования и найти тот тест, на котором Ваше решение работает неверно. После устранения ошибки решение можно отправить на повторную проверку.

Для задач «A», «B» нужно написать программу на C++/Java. Программа должна представлять собой один файл. Исходные данные нужно считывать через стандартный поток ввода (с клавиатуры), а ответ выводить в стандартный поток вывода (на экран). Программа должна использовать не более 500 мегабайт оперативной памяти и укладываться в ограничение по времени 1 секунда на тест. Ограничения и формат входных и выходных данных должны строго соответствовать условию задач. Гарантируется, что все тесты являются корректными.

Для задачи «C» нужно прислать только правильный ответ. Основная сложность задачи — подобрать подходящий инструментарий для решения. Вы можете использовать произвольные языки программирования, готовые утилиты или подбирать ответ вручную.

За все время тестирования разрешается отправить не более 200 решений суммарным объемом не более 10 мегабайт. Размер одного решения не должен превышать 200 килобайт.

При подведении результатов мы будем учитывать количество решенных задач, качество программного кода и количество штрафных попыток.

Задача A. Преобразование форматов

Рассматривается список поддиректорий и файлов некоторой директории в Linux, например:

```
.
./download_client.sh
./random1000_queries_sport.txt
./times.txt
./site
./site/site_kz_domains_random1000_2011-07-26.txt
./site/site_ru_domains_top1000_2011-07-27.txt
./site/site_by_domains_top1000_2011-07-27.txt
./site/kz
./site/kz/random1000
./site/kz/random1000/site_kz_random1000_2011-07-16.xml
./site/ru
./site/ru/random1000
```

Вам необходимо этот список из одного формата преобразовать в другой формат.

Формат входных данных

В первых двух строках даны названия входного и выходного формата. Обе строки — из множества «`find`», «`python`», «`acm1`», «`acm2`», «`acm3`», «`xml`». В последующих строках в формате, соответствующем входному формату из первой строки, описано дерево директорий. Ваша задача — в выходной файл вывести это же дерево директорий в выходном формате.

Гарантируется, что описываемое множество файлов и директорий описывает некоторое дерево с единственным выделенным корнем – корневой директорией. В частности, если перечислена какая-то поддиректория, то перечислены и все ее родительские директории вплоть до корневой, а также все ее поддиректории и файлы в них рекурсивно. Если есть элемент списка, у которого нет потомков в дереве, то это файл, иначе это директория. Соответственно, нет пустых директорий на нижнем уровне дерева. Все идентификаторы файлов и директорий — уникальные целые числа в диапазоне от 0 до $(2^{31}-1)$. Идентификатор корневой директории всегда равен 0. Идентификатор родительской вершины всегда меньше идентификатора дочерней вершины.

Имена файлов и папок — непустые строки, состоящие из строчных и заглавных символов латинского алфавита, цифр, символов тире, подчеркивания и точки.

Гарантируется, что размер входного файла и правильного выходного файла не превышает 250 килобайт.

Формат «`find`»

В первой строке число n — общее количество файлов и директорий. В каждой из следующих строк — полный путь к файлу или директории в обычном UNIX-формате, пробел и идентификатор файла или директории. Идентификаторы упорядочены по возрастанию.

```
13
. 0
./download_client.sh 1
./random1000_queries_sport.txt 2
./times.txt 3
./site 4
./site/site_kz_domains_random1000_2011-07-26.txt 5
./site/site_ru_domains_top1000_2011-07-27.txt 6
./site/site_by_domains_top1000_2011-07-27.txt 7
./site/kz 8
./site/kz/random1000 9
./site/kz/random1000/site_kz_random1000_2011-07-16.xml 10
./site/ru 11
./site/ru/random1000 12
```

Формат «python»

В первой строке число n — общее количество файлов и директорий. Далее дерево представлено в виде n строк, в каждой из которых описан один из файлов или одна из директорий и идентификатор. При этом описание каждого файла и каждой директории начинается после $4k$ пробелов, где k — уровень вложенности этого файла или директории. Уровень вложенности корневой директории — 0, директорий и файлов, находящихся непосредственно в корневой директории, — 1, и т.д. При этом описание самого файла или директории состоит только из имени этого файла или директории, а не полного пути от корневой директории. Кроме того, порядок строк в этом описании соответствует следующему порядку обхода дерева: сначала выписывается корневая вершина, затем выписывается ее первый ребенок (с наименьшим идентификатором), после чего рекурсивно выписывается его поддерево, затем выписывается второй ребенок с поддеревом, потом третий и т.д.

```
13
. 0
    download_client.sh 1
    random1000_queries_sport.txt 2
    times.txt 3
    site 4
        site_kz_domains_random1000_2011-07-26.txt 5
        site_ru_domains_top1000_2011-07-27.txt 6
        site_by_domains_top1000_2011-07-27.txt 7
        kz 8
            random1000 9
                site_kz_random1000_2011-07-16.xml 10
        ru 11
            random1000 12
```

Формат «acm1»

В первой строке число n — общее количество файлов и директорий. Далее n строк. В i-й строке название файла или директории, пробел и идентификатор. Идентификаторы упорядочены по возрастанию. Далее еще n строк. В i-й строке вначале записано число k — количество детей вершины с i-м по порядку идентификатором. Затем еще k чисел через пробел — идентификаторы детей в порядке возрастания.

```
13
. 0
download_client.sh 1
random1000_queries_sport.txt 2
times.txt 3
site 4
site_kz_domains_random1000_2011-07-26.txt 5
site_ru_domains_top1000_2011-07-27.txt 6
site_by_domains_top1000_2011-07-27.txt 7
kz 8
random1000 9
site_kz_random1000_2011-07-16.xml 10
ru 11
random1000 12
4 1 2 3 4
0
0
0
5 5 6 7 8 11
0
0
0
2 9 10
0
0
1 12
0
```

Формат «acm2»

В первой строке число n — общее количество файлов и директорий. Далее n строк. В i-й строке название файла или директории, пробел и идентификатор. Идентификаторы упорядочены по возрастанию. Далее еще n строк. В i-й строке записан идентификатор родителя вершины с i-м идентификатором или -1, если это корневая вершина.

```
13
. 0
download_client.sh 1
random1000_queries_sport.txt 2
times.txt 3
site 4
site_kz_domains_random1000_2011-07-26.txt 5
site_ru_domains_top1000_2011-07-27.txt 6
site_by_domains_top1000_2011-07-27.txt 7
kz 8
random1000 9
site_kz_random1000_2011-07-16.xml 10
ru 11
random1000 12
-1
0
0
0
0
4
4
4
4
8
8
4
11
```

Формат «acm3»

В первой строке число n — общее количество файлов и директорий. Далее n строк. В i-й строке название файла или директории, пробел и идентификатор. Идентификаторы упорядочены по возрастанию. Далее еще $(n-1)$ строка — описания ребер дерева. Каждое ребро описывается двумя идентификаторами вершин, записанными через пробел. Сначала записан меньший идентификатор, затем больший. Ребра выписаны в порядке возрастания первого идентификатора вершины, а при равенстве первых идентификаторов — в порядке возрастания второго идентификатора вершины.

```
13
. 0
download_client.sh 1
random1000_queries_sport.txt 2
times.txt 3
site 4
site_kz_domains_random1000_2011-07-26.txt 5
site_ru_domains_top1000_2011-07-27.txt 6
site_by_domains_top1000_2011-07-27.txt 7
kz 8
random1000 9
site_kz_random1000_2011-07-16.xml 10
ru 11
random1000 12
0 1
0 2
0 3
0 4
4 5
4 6
4 7
4 8
4 11
8 9
8 10
11 12
```

Формат «xml»

Файлы и директории в xml-формате перечисляются в следующем порядке обхода дерева: сначала перечисляется корневая директория, затем ее первый ребенок (с наименьшим идентификатором) и его поддерево, затем второй ребенок с поддеревом и т.д. Вложенные файлы и директории описываются вложенными тегами.

Директория описывается тегами

```
<dir name='site' id='4'>
...
</dir>
```

Файл описывается тегом

```
<file name='download_client.sh' id='1'/>
```

Каждый тег находится на отдельной строке. Сдвиг каждой строки составляет $2k$ пробелов, где k — уровень вложенности объекта. Уровень вложенности корневой директории 0, файлов и директорий, находящихся непосредственно в ней, — 1, и т.д. Закрывающий тег имеет тот же сдвиг, что соответствующий открывающий тег.

```
<dir name='.' id='0'>
  <file name='download_client.sh' id='1'/>
  <file name='random1000_queries_sport.txt' id='2'/>
  <file name='times.txt' id='3'/>
  <dir name='site' id='4'>
    <file name='site_kz_domains_random1000_2011-07-26.txt' id='5'/>
    <file name='site_ru_domains_top1000_2011-07-27.txt' id='6'/>
    <file name='site_by_domains_top1000_2011-07-27.txt' id='7'/>
    <dir name='kz' id='8'>
      <file name='random1000' id='9'/>
      <file name='site_kz_random1000_2011-07-16.xml' id='10'/>
    </dir>
    <dir name='ru' id='11'>
      <file name='random1000' id='12'/>
    </dir>
  </dir>
</dir>
```

Формат выходных данных

В выходной файл выведите список файлов в новом формате.

Пример

входные данные	выходные данные
find python 2 site 0 site/news 12	2 site 0 news 12

Задача B. Наибольшая общая подстрока

Даны K строк, нужно найти их наибольшую общую подстроку.

Формат входных данных

В первой строке записано целое число K ($1 \leq K \leq 10$). Далее записаны исходные K строк. Каждая строка состоит из не более чем 10 000 маленьких латинских букв.

Формат выходных данных

Выведите наибольшую общую подстроку.

Примеры

входные данные	выходные данные
3 abacaba mycabarchive acabistrue	cab

Задача C. Глобальные переменные

Дана программа на языке C++, необходимо найти и вывести список глобальных переменных.

Формат входных данных

Входные данные представляют собой архив, в котором содержится множество .cpp файлов. Каждый такой .cpp файл представляет собой синтаксически корректную программу на языке C++ (успешно компилируется при помощи компиляторов GNU C++ и Microsoft Visual C++).

Формат выходных данных

Для каждого файла из архива выведите в отдельной строке название этого файла и список глобальных переменных так, как это указано в примере.

Пример

выходные данные
000000.cpp ancestor ansv cost graph M N p qr query u 000001.cpp 000002.cpp 000003.cpp 000004.cpp 000005.cpp 000006.cpp 000007.cpp edge tree ...