

# Задания очного тура

Олимпиада по веб-программированию, Академия «1С-Битрикс», фирма «1С», 2019 год

## А. Туда-сюда

**Имя входного файла:** input.txt или стандартный поток ввода  
**Имя выходного файла:** output.txt или стандартный поток вывода

Николай Иванович еще не успел познакомиться с коллегами лично, но уже получил доступ в GIT-репозиторий “Аэропорт города N”. Через минуту пришли приглашение в таск-трекер и уведомление о новой задаче. Задача касалась разработки информационного табло с информацией о ближайших авиарейсах. Николай Иванович ознакомился с задачей. На вход приходили данные о часовом поясе в аэропорте отправления, времени вылета, часовом поясе в аэропорте прибытия и времени прибытия. Осталось посчитать время перелета.

### Входные данные

В первой строке целое положительное число  $n$  ( $1 \leq n \leq 10^4$ ) — количество рейсов.

В следующих  $n$ -строках описание рейсов, по одному в строке. Каждое описание задается четырьмя значениям:

1. местное время вылета в формате d.m.Y\_H:i:s;
2. часовой пояс вылета;
3. местное время прибытия в формате d.m.Y\_H:i:s;
4. часовой пояс прибытия.

### Выходные данные

Необходимо вывести  $n$  строк, в каждой строке одно число — время в полете в секундах.

### Пример

Входные данные	Результат работы
2 19.12.2018_19:10:00 -5 20.12.2018_12:25:00 3 19.12.2018_06:45:00 8 19.12.2018_10:40:00 1	33300 39300

## В. Я его слепила из того что было

**Имя входного файла:** sections.xml и products.xml

**Имя выходного файла:** output.xml

Сегодня Николай Иванович пришел на работу раньше обычного. Предстояло первый раз заниматься выгрузкой товаров из интернет-магазина. Николай Иванович был знаком с XML, но ему не доводилось собирать его из нескольких частей. На первый взгляд, все было просто, только каждый товар принадлежал нескольким разделам. Николай Иванович почесал голову и стал размышлять над формированием общего файла без потери данных.

### Входные данные

Входными данными являются два xml-файла:

1. section.xml следующей структуры:

```
<Разделы>
  <Раздел>
    <Ид>UUID раздела</Ид>
    <Наименование>Наименование раздела</Наименование>
  </Раздел>
</Разделы>
```

2. products.xml следующей структуры:

```
<Товары>
  <Товар>
    <Ид>UUID товара</Ид>
    <Наименование>Наименование товара</Наименование>
    <Артикул>Артикул товара</Артикул>
    <Разделы>
      <ИдРаздела>UUID раздела</ИдРаздела>
    </Разделы>
  </Товар>
</Товары>
```

Гарантируется, что разделов в файле section.xml всегда больше одного. Однако товаров в файле products.xml может и не быть.

Один товар может принадлежать нескольким разделам.

### Выходные данные

В файл output.xml требуется вывести объединенный XML-документ следующей структуры:

```
<ЭлементыКаталога>
```

```
  <Разделы>
```

```
    <Раздел>
```

```
      <Ид>UUID</Ид>
```

```
      <Наименование>Наименование раздела</Наименование>
```

```

<Товары>
  <Товар>
    <Ид>UUID товара</Ид>
    <Наименование>Наименование товара</Наименование>
    <Артикул>Артикул товара</Артикул>
  </Товар>
</Товары>
</Раздел>
</Разделы>
</ЭлементыКаталога>

```

## Пример

Входные данные	Результат работы
<b>products.xml</b> <?xml version="1.0" encoding="UTF-8"?> <Товары/>  <b>sections.xml</b> <?xml version="1.0" encoding="UTF-8"?> <Разделы> <Раздел> <Ид>0869ff5d-d492-4a84-9c8b-1b2b8686e071</Ид> <Наименование>Смартфоны</Наименование> </Раздел> <Раздел> <Ид>8bc194f3-7fcc-4b25-bf19-2264a241a26b</Ид> <Наименование>Бытовая техника</Наименование> </Раздел> </Разделы>	<b>output.xml</b> <?xml version="1.0" encoding="UTF-8"?> <ЭлементыКаталога> <Разделы> <Раздел> <Ид>0869ff5d-d492-4a84-9c8b-1b2b8686e071</Ид> <Наименование>Смартфоны</Наименование> <Товары/> </Раздел> <Раздел> <Ид>8bc194f3-7fcc-4b25-bf19-2264a241a26b</Ид> <Наименование>Бытовая техника</Наименование> <Товары/> </Раздел> </Разделы> </ЭлементыКаталога>
<b>products.xml</b> <?xml version="1.0" encoding="UTF-8"?> <Товары> <Товар> <Ид>0e8d7c95-7a1b-4ae6-b758-01002a0200a9</Ид> <Наименование>iPhone 8</Наименование> <Артикул>2342</Артикул> <Разделы> <ИдРаздела>0869ff5d-d492-4a84-9c8b-1b2b8686e071</ИдРаздела> </Разделы> </Товар> </Товары>  <b>sections.xml</b> <?xml version="1.0" encoding="UTF-8"?> <Разделы> <Раздел> <Ид>0869ff5d-d492-4a84-9c8b-1b2b8686e071</Ид> <Наименование>Телефоны</Наименование> </Раздел> </Разделы>	<b>output.xml</b> <?xml version="1.0" encoding="UTF-8"?> <ЭлементыКаталога> <Разделы> <Раздел> <Ид>0869ff5d-d492-4a84-9c8b-1b2b8686e071</Ид> <Наименование>Телефоны</Наименование> <Товары> <Товар> <Ид>0e8d7c95-7a1b-4ae6-b758-01002a0200a9</Ид> <Наименование>iPhone 8</Наименование> <Артикул>2342</Артикул> </Товар> </Товары> </Раздел> </Разделы> </ЭлементыКаталога>

## С. Точно в цель

**Имя входного файла:** input.txt или стандартный поток ввода

**Имя выходного файла:** output.png

Не успел Николай Иванович отправить свой код на проверку руководителю, как пришло уведомление о новой задаче. Вот что значит высоконагруженный проект, подумал Николай Иванович. Не успеешь сдать задачу, как приходит следующая. Необходимо было проанализировать целевые действия пользователей на сайте и построить воронку продаж. Николай Иванович предположил, что задача связана с интернет-маркетингом и веб-аналитикой и пошел искать что такое “Воронка продаж” и “Целевые действия”.

Целевое действие - действие пользователя на сайте, которое приближает его к конверсии в покупателя. Например: “посещение карточки товара”, “добавление товара в корзину”, “оформление заказа”.

Конверсионный путь - строгая последовательность из целевых действий, которые приводят к конверсии посетителя в покупателя.

Воронка продаж - график напоминающего перевернутую пирамиду. Каждая часть соответствует целевому действию. Верхняя широкая часть показывает сколько посетителей начали свой конверсионный путь, нижняя -- сколько его завершили оформлением заказа.



### Входные данные

В первой строке целое положительное число  $n$  ( $1 \leq n \leq 10$ ) — количество целей.

В следующих  $n$ -строках содержатся идентификаторы целей (строка) в порядке конверсионного пути.

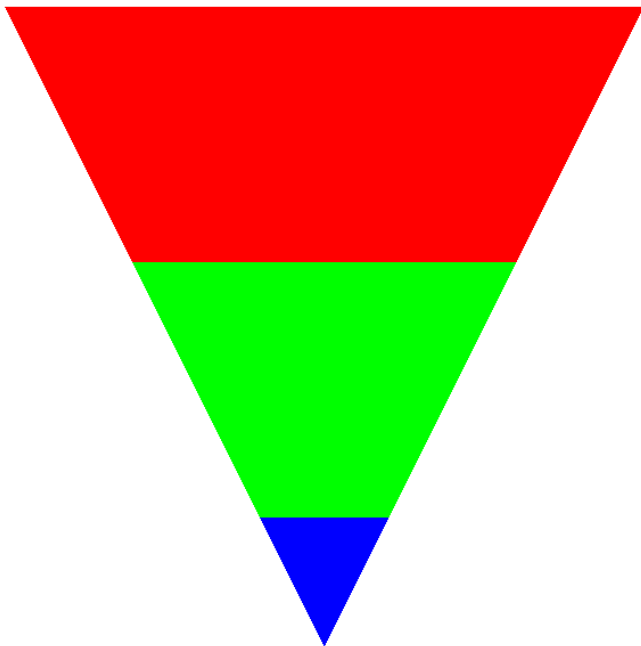
Далее идет целое положительное число  $m$  ( $1 \leq m \leq 100$ ) — количество событий.

В следующих m-строках содержатся события, по одному в каждой строке. Каждое событие описывается двумя значениями идентификатор пользователя (строка) и идентификатор выполненной цели (строка).

### Выходные данные

В output.png вывести изображение размером 600x600 пикселей, каждый сегмент воронки покрасить в разные цвета.

### Пример

Входные данные	Результат работы
3 DETAIL CART CHECKOUT 9 user1 DETAIL user2 DETAIL user1 CART user3 DETAIL user3 CART user1 CHECKOUT user4 DETAIL user5 DETAIL user5 CART	

### D. Время = деньги

**Имя входного файла:** input.html  
**Имя выходного файла:** output.html

Заканчивался первая неделя на новом рабочем месте. Николай Иванович был доволен собой, успел найти общий язык с новыми коллегами и въедливым руководителем. Осталось закрыть последнюю задачу спринта и начнутся заслуженные выходные. Задача была непростой, но крайне интересной. Нужно увеличить скорость загрузки страниц на сайте. Проконсультировавшись с коллегами Николай Иванович узнал, что для увеличения скорости нужно:

1. Сократить html.

- a. теги code, pre, textarea, style и script оставляем без изменений;
  - b. больше одного пробела подряд заменяем на один пробел;
  - c. удалить пробелы, переносы строк и табуляцию вне тегов;
  - d. удалить комментарии, если они не начинается со специального слова skip-delete.
2. Перенести подключение js в конец html-страницы, если у тега нет атрибута data-skip-moving со значение true.

## Входные данные

Html-страница с именем input.html.

## Выходные данные

Оптимизированная страница с именем output.html.

## Пример

input.html

```
<!DOCTYPE HTML>
<html lang="ru-RU">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

  <title> Заголовок страницы с пробелами </title>

  <script src="/move_to_bottom1.js"></script>
  <script async src="/move_to_bottom2.js"></script>

  <script data-skip-moving="true" src="/script1.js"></script>
  <script async data-skip-moving="true" src="/script2.js"></script>
  <script src="/script3.js" data-skip-moving="true"></script>

  <script data-skip-moving="true">
    function f1() { /* DON'T MOVE TO BOTTOM */ }
  </script>

  <script data-skip-moving="false">
    function f1() { /* MOVE TO BOTTOM */ }
  </script>

  <link rel="stylesheet" href="/style1.css">

</head>
<body>

<script>
  function f2() { /* MOVE TO BOTTOM */ }

  /* DO NOT DELETE WHITESPACES */
```

</script>

<style>

```
body {
    background: wheat;
}
```

</style>

<h1>title with <!-- COMMENT IN TEXT --> whitespaces </h1>

<!--skip-delete SKIPPED COMMENT -->

<div class="container">

<div class="row test">

<div class="col-12">

<span>Текст с пробелами и  
переносами строк</span>

</div>

<br>

<label for="t">

<textarea name="t" id="t" cols="30" rows="10">

Внутри тега textarea пробелы и переносы строк  
не убираем

</textarea>

</label>

</div>

</div>

<pre>

```
тег
pre
```

</pre>

<pre class="with\_class">

```
тег
pre
с
классом
```

</pre>

<code>

```
тег
code
```

</code>

<code class="with\_class">

```
тег
code
с
классом
```

</code>

</body>

</html>

output.html

```
<!DOCTYPE HTML><html lang="ru-RU"><head><meta http-equiv="Content-Type" content="text/html; charset=utf-8" /><title> Заголовок страницы с пробелами </title><script data-skip-moving="true" src="/script1.js"></script><script async data-skip-moving="true" src="/script2.js"></script><script src="/script3.js" data-skip-moving="true"></script><script data-skip-moving="true">
    function f1() { /* DON'T MOVE TO BOTTOM */ }
</script><link rel="stylesheet" href="/style1.css"></head><body><style>
    body {
        background: wheat;
    }
</style><h1>title with whitespaces </h1><!--skip-delete SKIPPED COMMENT --><div class="container"><div class="row test"><div class="col-12"><span>Текст с пробелами и переносами строк</span></div><br><label for="t"><textarea name="t" id="t" cols="30" rows="10">
Внутри тега textarea пробелы и переносы строк
не убираем
        </textarea></label></div></div><pre>
    тер
    пре
</pre><pre class="with_class">
    тер
    пре
    с
    классом
</pre><code>
    тер
    code
</code><code class="with_class">
    тер
    code
    с
    классом
</code></body><script src="/move_to_bottom1.js"></script><script async src="/move_to_bottom2.js"></script><script data-skip-moving="false">
    function f1() { /* MOVE TO BOTTOM */ }
</script><script>
    function f2() { /* MOVE TO BOTTOM */ }

    /* DO NOT DELETE WHITESPACES */

</script></html>
```