Тестовое задание на должность разработчика

Имеется набор изображений **разной** высоты и ширины. Изображения могут располагаться слева направо, образуя строку, или сверху вниз, образуя колонку. Строки, в свою очередь, кроме изображений могут содержать колонки, а колонки - строки. Такую структуру изображений, строк и колонок далее будем называть “*деревом изображений*”. Если все элементы строки (изображения и колонки) приведены к одной высоте, а элементы колонки (изображения и строки) - к одной ширине **с сохранением пропорций**, то такое дерево будем называть “*раскадровкой изображений*”.

Необходимо написать программу, на вход которой поступает дерево изображений и число пикселей ω, на выход - изображение раскадровки шириной ω, при этом, выполняется **одно из** условий (от первого и самого лёгкого до последнего самого сложного):

1. Дерево состоит только из одной строки с произвольным количеством изображений без колонок.
2. Дерево изображений содержит произвольное количество изображений, строк и колонок на произвольных местах.
3. Второе условие выполняется и на вход поступают дополнительные четыре параметра, учитывающиеся при формировании раскадровки: отступ от изображения сверху, справа, снизу и слева[3].

Претенденты, выполнившие более сложное задание, имеют приоритет над выполнившими более лёгкое. Также учитываются стиль написания и приёмы, используемые при решении. Если что-то не получается сделать - это нормально, можно выполнить часть задания или воспользоваться литературой и электронными ресурсами. Разрешено использовать любой язык программирования.

Пример реализации лёгкого задания (JavaScript):

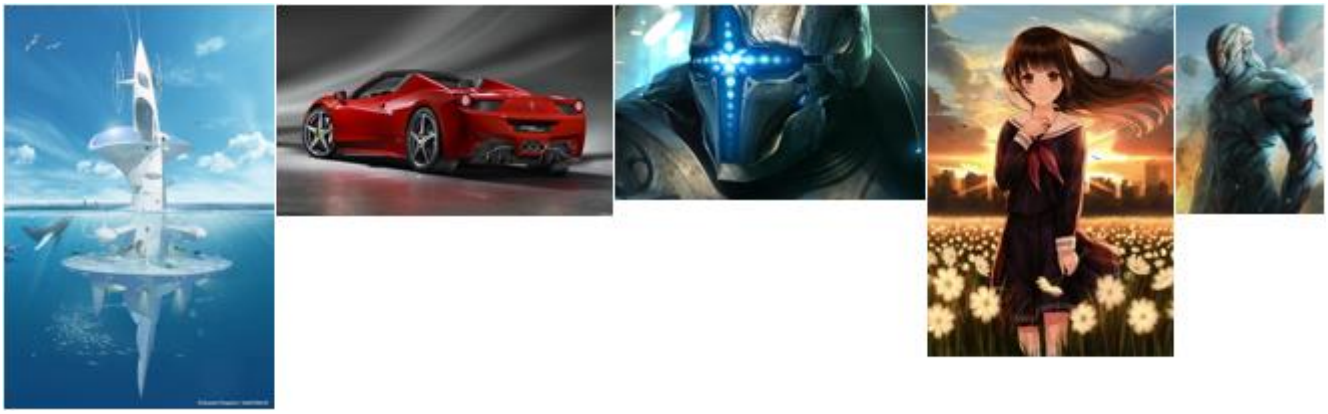
Строку можно представить объектом типа `row`, колонку - `column`. У обоих из них добавить метод `add(frame)`, который возвращает указатель на самого себя:

```
row.add(frame); // Где объект frame может быть изображением Image или столбцом column.
column.add(frame); // Где объект frame может быть изображением Image или строкой row.
```

Тогда код создания дерева для раскадровки в одну строку для **n** изображений будет иметь такой вид:

```
var r = new row(); // Дерево раскадровки (DOM с набором картинок).
r.add(img(“1.jpg”)).add(img(“2.jpg”)) ... .add(img(“n.jpg”)); // См. примечание[1] ниже.
```

Визуально структуру **r** можно представить так:



Для формирования изображения необходимо написать, метод `drawStoryboard(frame, args)`, где `frame` - корневой узел дерева, содержимое которого необходимо вывести на экран в виде раскадровки, а `args` – набор параметров требуемых по заданию (ширина и отступы, в зависимости от выбранной сложности).
В итоге общий вид приложения будет иметь вид:

```
var r = new row(); // Дерево раскадровки (DOM с набором картинок).
r.add(img("1.jpg")).add(img("2.jpg")) ... .add(img("n.jpg")); // Примечание[2].
// Формирование из дерева изображений раскадровки и вывод на экран:
drawStoryboard(r, {
       width: 600, // Требуемая ширина раскадровки ω.
});
```

После выполнения этого кода должны получить изображения раскадровки общей шириной в 600 пикселей:

Так выглядит реализация простого варианта задания.

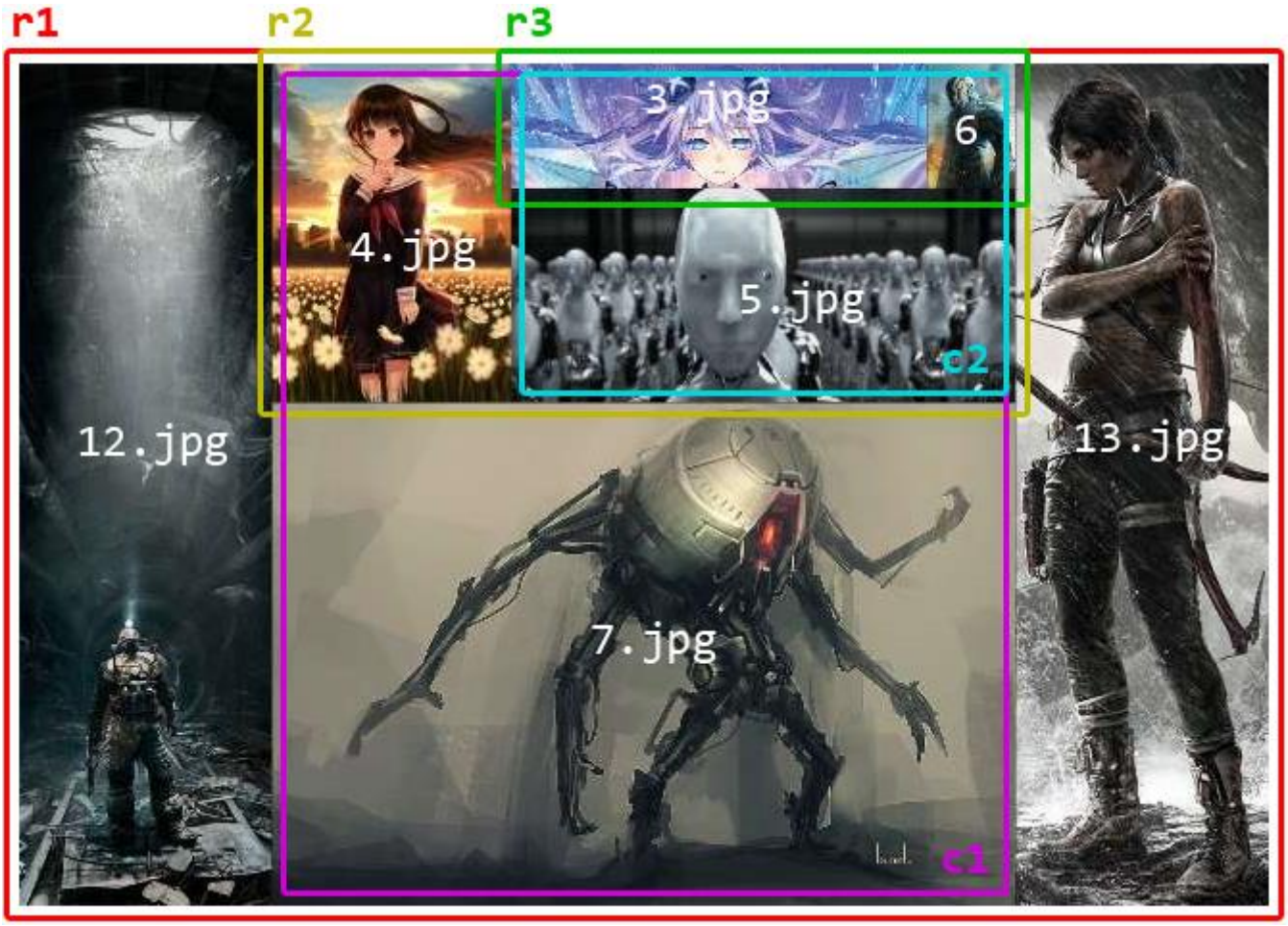
[1] **Для ввода входных данных** в программе **необязательно реализовывать интерфейс** пользователя. Можно ограничиться описанным выше определением в коде (при необходимости мы вручную подставим свои данные).
[2] `img("файл")` – в данном случае понимается как функция, возвращающая объект с изображением из файла. В JavaScript удобно использовать `HTMLCanvasElement` или `Image`.

Формирование раскадровок, на реализацию которых обращается наибольшее внимание

Для создания более полезной раскадровки можно воспользоваться таким деревом:

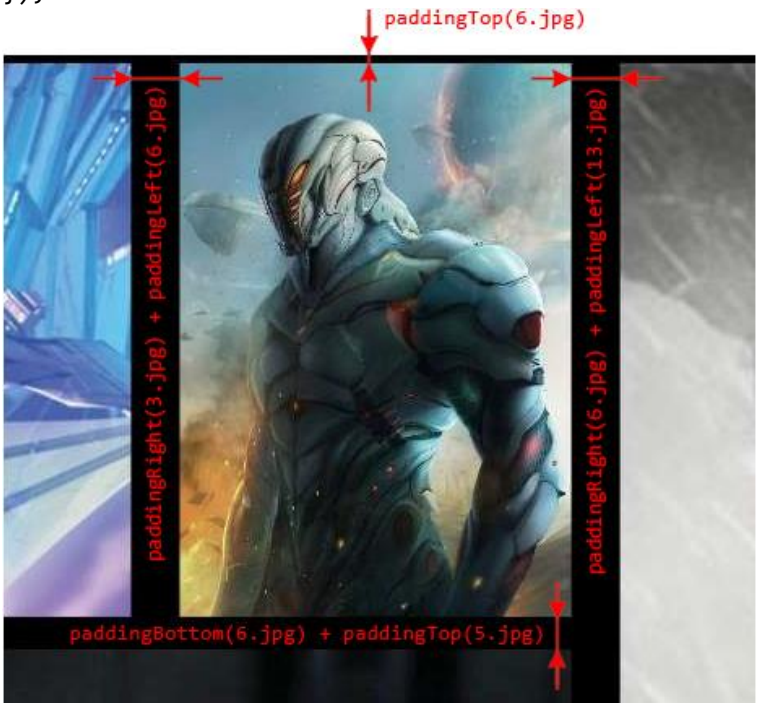
```
var r1 = new row(),
    c1 = new column(),
    r2 = new row(),
    c2 = new column(),
    r3 = new row();

r1.add(img("12.jpg")).add(c1).add(img("13.jpg"));
c1.add(r2).add(img("7.jpg"));
r2.add(img("4.jpg")).add(c2);
c2.add(r3).add(img("5.jpg"));
r3.add(img("3.jpg")).add(img("6.jpg"));

drawStoryboard(r1, {
       width: 650, // Требуемая ширина раскадровки ω.
});
```

Результат раскадровки представлен на рисунке ниже:



³ Для выполнения самого сложного задания необходимо добавить еще четыре параметра с отступами от изображения (эти отступы применяются к каждому изображению отдельно):

```
drawStoryboard(r1, {
	width: 650, // Требуемая ширина раскадровки ω.
	paddingTop: 5, // Отступы для задания со сложным условием:
	paddingRight: 10, // Отступ справа (в пикселях).
	paddingBottom: 15, // Отступ снизу.
	paddingLeft: 20 // Отступ слева.
});
```



На выход приложения достаточно вывести изображение с раскадровкой:

Дополнительный пример:

```
var r1 = new row(),
    r2 = new row(),
    r3 = new row(),
    r4 = new row(),
    r5 = new row(),
    r6 = new row(),
    r7 = new row(),
    c1 = new column(),
    c2 = new column(),
    c3 = new column(),
    c4 = new column(),
    c5 = new column();

r1.add(img("1.jpg")).add(c1).add(c2).add(img("19.jpg"));
c2.add(img("17.jpg")).add(img("18.jpg"));
c1.add(r2).add(r3).add(img("16.jpg"));
r2.add(c3).add(img("5.jpg"));
c3.add(r4).add(img("4.jpg"));
r4.add(img("2.jpg")).add(img("3.jpg"));
r3.add(img("6.jpg")).add(c4).add(c5);
c4.add(r5).add(img("10.jpg")).add(r6);
r5.add(img("7.jpg")).add(img("8.jpg")).add(img("9.jpg"));
r6.add(img("11.jpg")).add(img("12.jpg"));
c5.add(r7).add(img("15.jpg"));
r7.add(img("13.jpg")).add(img("14.jpg"));

drawStoryboard(r1, {width: 3650, // Требуемая ширина раскадровки ω.
       paddingTop: 5, paddingRight: 10, // Отступ справа.
       paddingBottom: 15, paddingLeft: 20 // Отступ слева.
});
```


©Jacques Rougerie / SeaOrbiter®
SHIKANA
YUKARI