1. JavaScript: Image Cloning

Create an Image class that supports image cloning. Multiple tools make copy-pasting images possible by cloning them.

Implementation of class *Size* is already provided.

1. This class has a constructor *Size(width, height)* to set the width and height of the *Size* object created.

Implement the *Image* class with the following constructor and methods:

1. The constructor *Image(String url, Size size)* sets the url and size of the image object created.
2. The method *getUrl()* returns the *url*.
3. The method *setUrl(url)* updates the *url*.
4. The method *setSize(width, height)* updates the *width* and *height* values of the size property.
5. The method *getSize()* returns the size of the image as a *Size* object.
6. The method *cloneImage()* returns a clone of the current image. Return a new *Image* instance with the same properties (*url* and *size*) as the current object.

The locked stub code validates the correctness of the *Image* class implementation by performing the following operations on the images:

- *Clone Id*: This operation creates a clone of the image.
- *UpdateUrl Id newUrl*: This operation updates the *url* of the image.
- *UpdateSize Id newWidth newHeight*: This operation updates the *size* of the image.

After performing all the operations, the locked stub code prints the url and size of each image.

▼ Input Format For Custom Testing

The first line contains an integer n, denoting the total number of images to be created initially.
Each of the next n lines contains 3 values - string url, number width, and number height for construction of n Image objects.
The next line contains the value of *numberOfOperations*, the total number of operations to be performed.
Each of the next *numberOfOperations* lines contains one of the three operations listed above.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN                                   Function
-----                                   --------
2                                   →   2 images to be created initially
hackerrank.com/image1 100 100       →   url = 'hackerrank.com/image1', width = 100, height = 100
hackerrank.com/image2 200 200       →   url = 'hackerrank.com/image2', width = 200, height = 200
3                                   →   numberOfOperations = 3
Clone 1                             →   clones 1st image object
UpdateUrl 1 hackerrank.com/image3   →   updates 1st image object's url to 'hackerrank.com/image3'
UpdateSize 2 300 400                →   updates 2nd images object's width to 300 and height to 400
```

Sample Output

```
hackerrank.com/image3 100 100
hackerrank.com/image2 300 400
hackerrank.com/image1 100 100
```

Explanation

The first line of input contains integer 2, which denotes 2 images that need to be created initially. 2 images are created with the inputs from the 2^{nd} and 3^{rd} lines. The next contains number 3, the number of operations to be performed. The next operation clones the images with id 1 (i.e. the first image object). The next line operation updates the url of the image with id 1 to 'hackerrank.com/image3'. The next operation updates the width and height of the image with id 2 to 300 and 400 respectively.