Name: Baihan Lin

UWID: sunnylin, 1360521

Section: BC

1. Describe your approach to implementing the hash table. (separate arrays for keys and values vs one array with objects that represent pairs.)

To implement my hash table, I decided to use one array with objects that represent pairs. I created a private inner class called HashPair inside ColorHash, and each HashPair stores a pair of ColorKey type key and a long type value.

2. If you implemented any methods other than those in the specification, describe them briefly here.

For ColorHash.class:

**int twiceAndPrime(int tableSize)** is a private helper method to support resize, taking @param tableSize: old table size that needs to be resized and @return newTableSize: the new table size which is at least twice as big the old table size and has to be a prime number.

**int[] doProbing(ColorKey key)** is a private helper method to support colorHashPut, increment, colorHashGet, getCount, with purpose of retrieving the insert location of ColorKey while summing number of collisions based on linear or quadratic probing based on collision resolution method from constructor input. It takes @param key: the ColorKey to insert in the hash table and @return an integer array storing the index of insert location and number of collisions during the probing.

**boolean ifRehash()** is a private helper method to support colorHashPut, increment, with a purpose of determining whether rehash load factor is exceeded and rehash is necessary. It @return a boolean to indicate the necessity of rehashing. If necessary to rehash, it does rehashing.

**HashPair[] getHashTable()** is a private helper method to support resize. It @return the raw HashPair[] format of hash table of the given ColorHash.

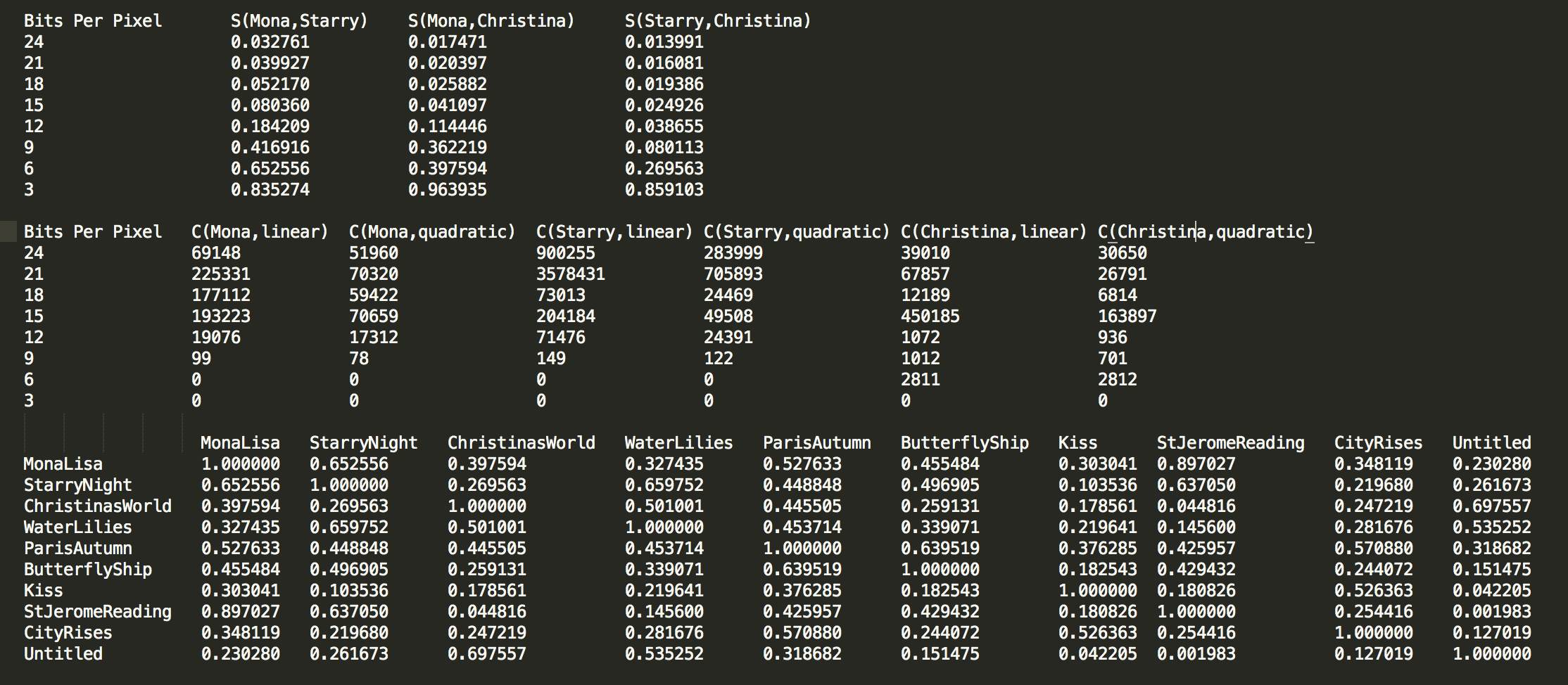
The methods specified inside HashPair class is described above in question 1.

For FeatureVector.class:

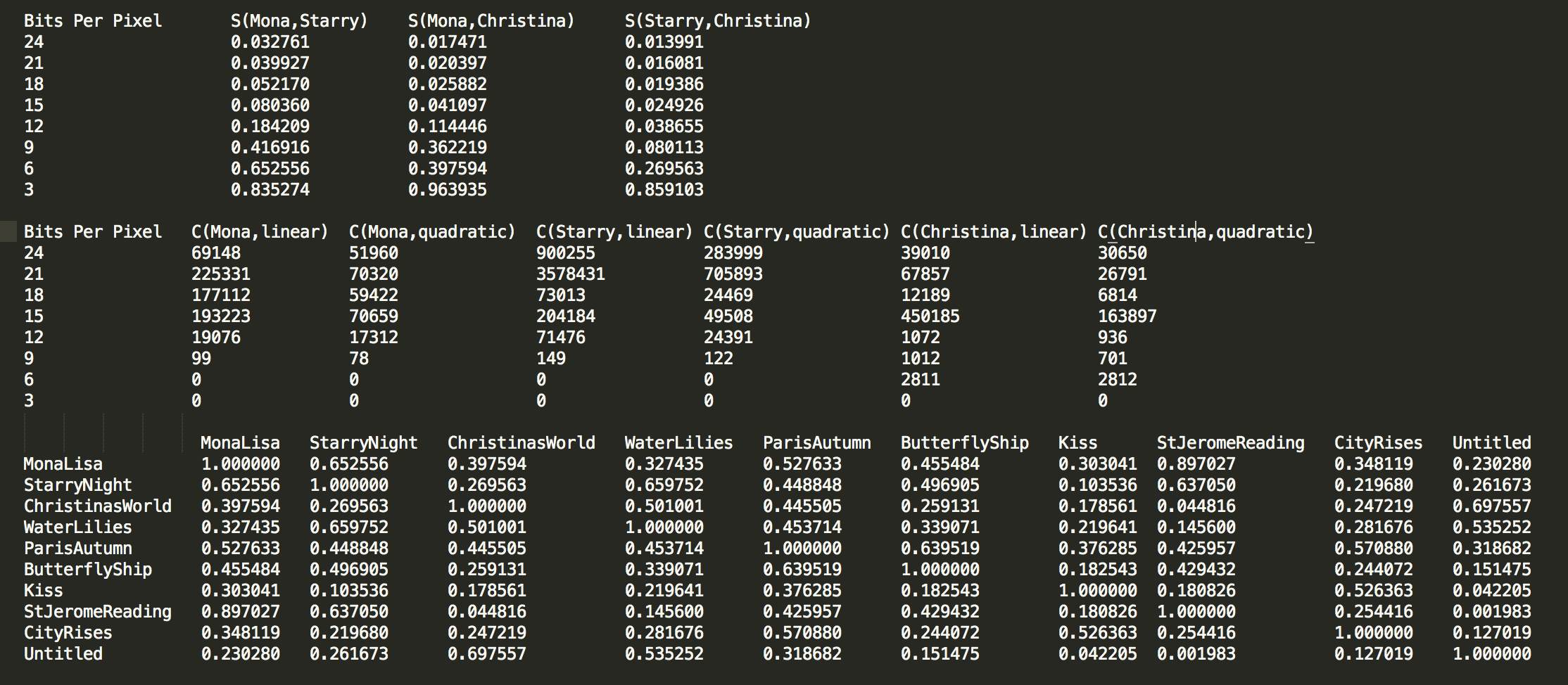
For ComparePaintings.class:

3. When you use 6 bits per pixel, how many black pixels are there in the Mona Lisa image? (These are the pixels whose ColorKey bits value equals 0.)

4. Provide a copy of the table of counts produced by your collisionTests method.

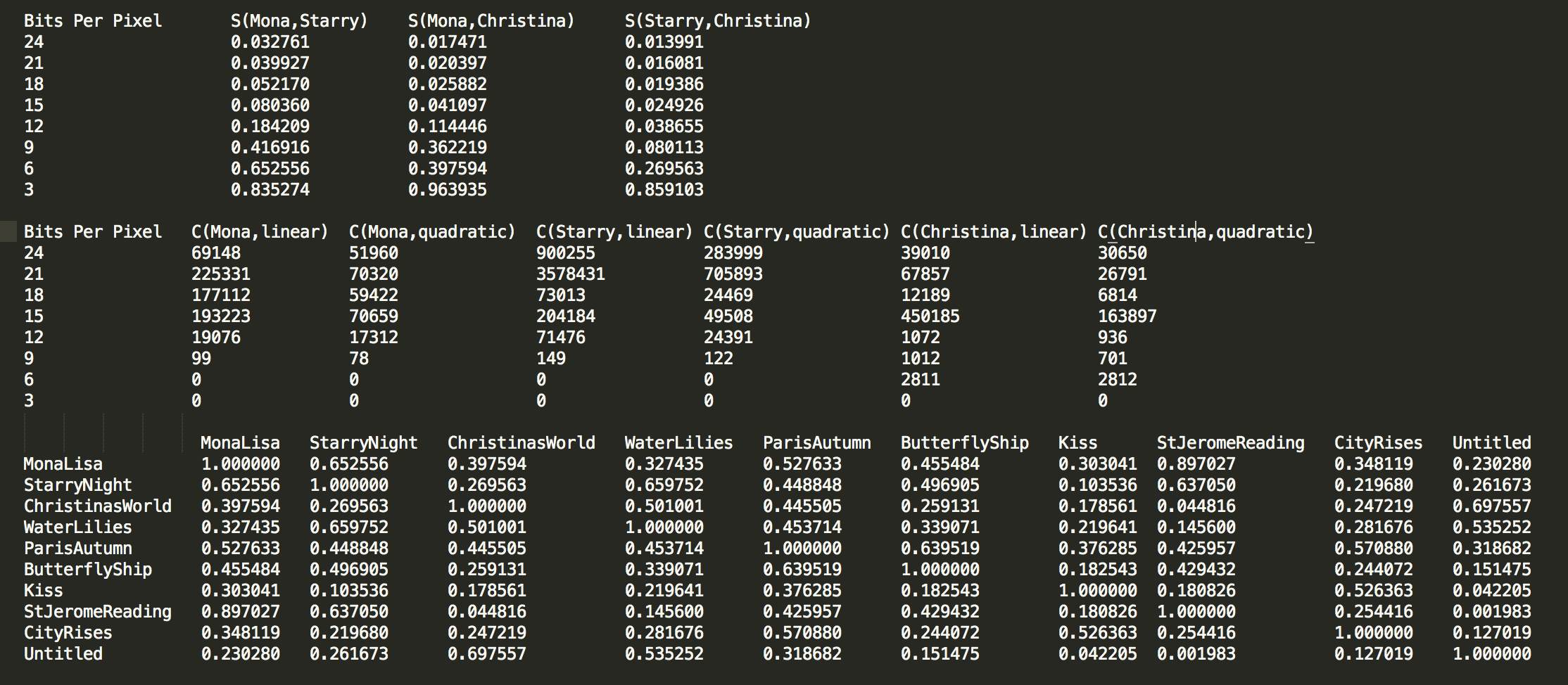


5. Provide a copy of the table of similarity values by your fullSimilarityTests method.



6. Examine the hashCode method of class ColorKey. What types of images might tend to cause lots of collisions relative to other images?

7. Extra Credit: Cosine similarity of 10 different images

Table of cosine similarity based on output from Java Eclipse console:

As we can see from the table, C(MonaLisa, StJeromeReading) = 0.897027 is the most similar image pair with the maximum cosine similarity. Visually speaking, “Mona Lisa” and “St. Jerome Reading” indeed look similar.

**Image Summary:**

1. “Mona Lisa” by Leonardo Da Vinci

<https://courses.cs.washington.edu/courses/cse373/16au/A/A3/>

1. “Starry Night” by Vincent Van Gogh

<https://courses.cs.washington.edu/courses/cse373/16au/A/A3/>

1. “Christina’s World” by Andrew Wyeth

<https://courses.cs.washington.edu/courses/cse373/16au/A/A3/>

1. “Water Lilies” by Claude Monet

<http://www.jenniferlynking.com/wp-content/uploads/2013/09/waterlilies_0410_-367.jpg>

1. “Paris Autumn” by Dmitry Spiros

<https://img1.etsystatic.com/074/0/7777713/il_570xN.823315899_kot1.jpg>

1. “Butterfly Ship” by Salvador Dali

<http://www.wallpaperawesome.com/wallpapers-awesome/wallpapers-famous-painting-artist-painter-brush-oil-on-canvas-awesome/wallpaper-salvador-dali-1.jpg>

1. “Kiss” by Gustav Klimt

<https://uploads8.wikiart.org/images/gustav-klimt/the-kiss-1908(1).jpg>

1. “St. Jerome Reading” by Georges de La Tour

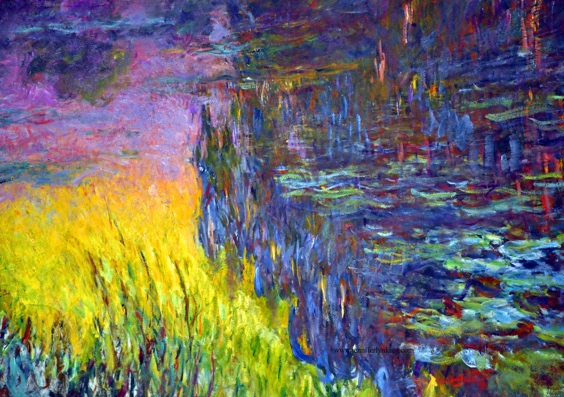
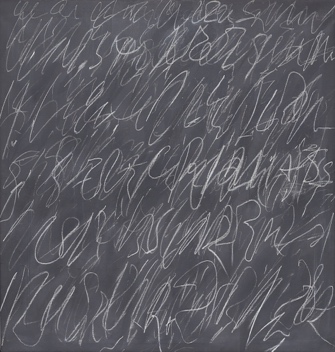
<http://www.abcgallery.com/L/latour/latour66.JPG>

1. “City Rises” by Boccioni

<http://exhibitions.guggenheim.org/futurism/content/small/futurism_heroic_boccioni_the_city_rises.jpg>

1. “Untitled” by Cy Twombly

<https://news.artnet.com/app/news-upload/2015/02/2015-feb-11-cy-twombly-untitled-new-york-city-1970-christies-evening-auction.jpg>

**4 5 6**

**7 8 9 10**

**1 2 3**