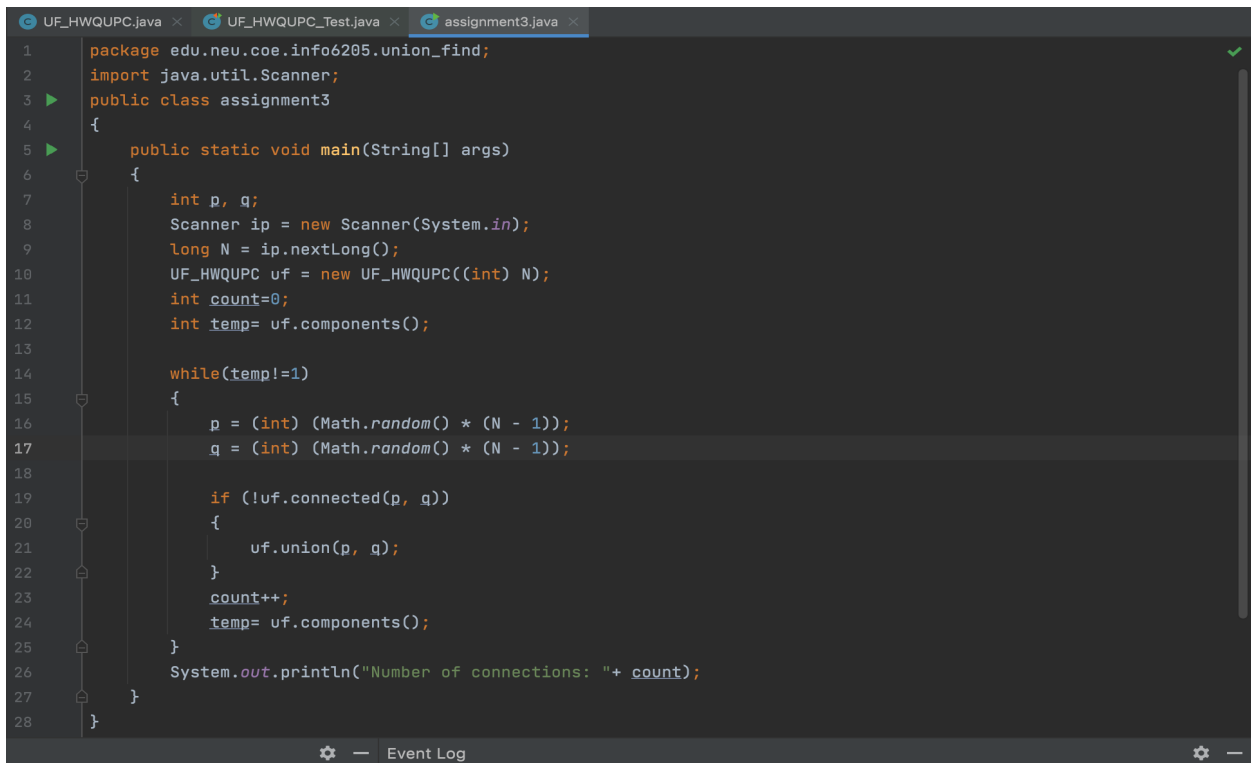Name: Namrata Ruchandani
NU ID : 002125637

➔ Task done in this assignment:

1. In the find, I'm checking pathCompression first. If yes then I'm calling dopathCompression by passing root, and updating root as parent of root. If there is no pathCompression, I'm just pointing to the parent of root not grandparent. Finally, returning root.
2. In the mergeComponents, I'm comparing the height of two roots to be merged. Making root with larger size as parent of root with smaller size. If the sizes are similar, attach them and increase height by 1.
3. In the doPathCompression, I'm pointing to grandparent instead of parent, and thus doing pathCompression.
4. In the file name assignment3.java, I took input by using scanner command and then made an object **uf** of UF_HWQUPC class. I stored the value of the number of components after each union in temp. I made a while loop with a condition when temp is not equal to 1, as when all components are connected, the number of components will be one. Generated p and q random numbers and then checked if connected or not. If not, I did union. Number of times random pairs are generated is counted by count. Hence, printing count as the number of pairs generated.

➔ The screenshot of main method (or part 2 of assignment) assignment3.java code compilation and output

```java
package edu.neu.coe.info6205.union_find;
import java.util.Scanner;
public class assignment3
{
    public static void main(String[] args)
    {
        int p, q;
        Scanner ip = new Scanner(System.in);
        long N = ip.nextLong();
        UF_HWQUPC uf = new UF_HWQUPC((int) N);
        int count=0;
        int temp= uf.components();

        while(temp!=1)
        {
            p = (int) (Math.random() * (N - 1));
            q = (int) (Math.random() * (N - 1));

            if (!uf.connected(p, q))
            {
                uf.union(p, q);
            }
            count++;
            temp= uf.components();
        }
        System.out.println("Number of connections: "+ count);
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
240
Number of connections: 484

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   🔧 Build   ≋ Dependencies

Build completed successfully in 2 sec, 383 ms (12 minutes ago)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
400
Number of connections: 806

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   🔧 Build   ≋ Dependencies

All files are up-to-date (moments ago)

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
676
Number of connections: 973

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   🔧 Build   ≋ Dependencies

All files are up-to-date (moments ago)

```
Run:      assignment3  ×

/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
803
Number of connections: 1158

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   ⚒ Build   ☰ Dependencies

All files are up-to-date (moments ago)

```
Run:      assignment3  ×

/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
921
Number of connections: 1566

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   ⚒ Build   ☰ Dependencies

All files are up-to-date (moments ago)

```
Run:      assignment3  ×

/Library/Java/JavaVirtualMachines/jdk1.8.0_301.jdk/Contents/Home/bin/java ...
1000
Number of connections: 1838

Process finished with exit code 0
```

Git   ▶ Run   ☰ TODO   ❶ Problems   ⊵ Terminal   ⚒ Build   ☰ Dependencies
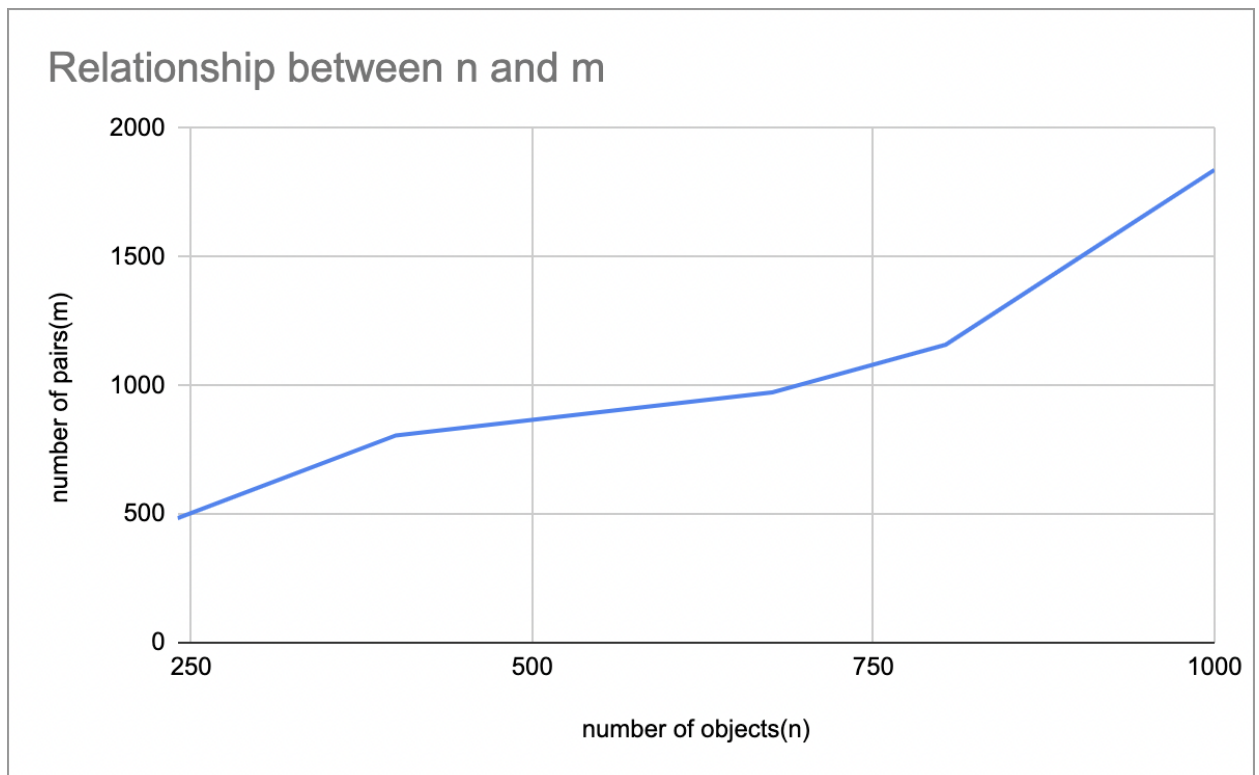
Build completed successfully in 2 sec, 204 ms (moments ago)

→ The link for file Assignment2.java :
→ Conclusion about number of objects(n) and number of pairs(m)
  Ideally, there should be n-1 connections to make one component in the end. But, as we
  are generating random pairs then repeated cases can occur. This is justified after seeing
  that the output of the number of pairs is more than n-1. The graph is linear showing the
  relation is **n + c,** where c is constant. I think, constant is added because of repeated
  pairs generated.

A	B
number of objects(n)	number of pairs (m)
240	484
400	806
676	973
803	1158
921	1566
1000	1838

Relationship between n and m

➔ I pushed the commands in the git and attached the link for the repository.
https://github.com/Namrata2108/INFO6205/tree/Fall2021/src/main/java/edu/neu/coe/info 6205

➔ Below is the screenshot of successful test cases run