CSC201 Data Structures
Fall 2020
Professor Burg
Program 3 -- Hashing
Analysis of Hash Table Implementation

General Instructions:
Each of you should do your own writeup of the results of your program tests. Your writeup should include:

- Tables for three tests cases. Fill out the tables below. **Notice that the tables for quadratic probing and double hashing are the same, and the tables for separate chaining are a little different.**
- Based on a comparison of your test cases, write a paragraph (at least) giving an overview explanation of the results.
- Based on discussing the results of your programs, write a paragraph (at least) comparing the results of the three methods -- not just raw numbers, but an overview of the differences in efficiency and your speculation as to why the methods compare as they do. What are the limiting factors in each of the methods? What parts of the code could be influencing the run-time?

The code for getting the program's runtime is given at the end of this document.

NOTE: In your program, print out each piece of information listed in the tables below, and label the information clearly.

Method Implemented:
File Name:
Dimensions:
Initial Table Size:
etc.

If you can't get the program to run:
If you are not able to get your program running, submit it anyway. Then your writeup should include:

- a description of what you were able to get running
- a description of what you were not able to get running
- the best explanation you can give for what you think is wrong

How to Submit:
Submit your writeup as a .docx or .pdf file. Submit this writeup at the same place on Canvas where you submit your Java code. Zip all the files together into one .zip file.

DATA GATHERING FOR QUADRATIC PROBING AND DOUBLE HASHING

Test Case 1	
file name (with pixel dimensions)	Ollie200x200.raw
initial table size	1001
# collisions	
# rehashes	
final table size	
runtime	

Test Case 2	
file name (with pixel dimensions)	Ollie200x200.raw
initial table size	101
# collisions	
# rehashes	
final table size	
runtime	

Test Case 3	
file name (with pixel dimensions)	either Hibiscus400x300 or LadyBugs800x600
initial table size	pixel dimensions/1000
# collisions	
# rehashes	
final table size	
runtime	

DATA GATHERING FOR SEPARATE CHAINING

Test Case 1	
file name (with pixel dimensions)	Ollie200x200.raw
initial table size	1001
# collisions	
# steps taken searching lists for RGB	
lengths of first 100 Freq lists in the hash table	"see program output"
runtime	

Test Case 2	
file name (with pixel dimensions)	Ollie200x200.raw
initial table size	101
# collisions	
# steps taken searching lists for RGB	
lengths of first 100 Freq lists in the hash table	"see program output"
runtime	

Test Case 3	
file name (with pixel dimensions)	either Hibiscus400x300 or LadyBugs800x600
initial table size	pixel dimensions/1000
# collisions	
# steps taken searching lists for RGB	
lengths of first 100 Freq lists in the hash table	"see program output"
runtime	

CODE FOR GETTING PROGRAM RUNTIME
Please put this code into your main method

```
import java.util.concurrent.TimeUnit;

// Program to calculate execution time or elapsed time in Java
class Main
{
	public static void main(String[] args) throws InterruptedException {

		long startTime = System.nanoTime();

		// ... the code being measured starts ...

		// sleep for 5 seconds
		TimeUnit.SECONDS.sleep(5);

		// ... the code being measured ends ...

		long endTime = System.nanoTime();

		long timeElapsed = endTime - startTime;

		System.out.println("Execution time in nanoseconds: " + timeElapsed);
		System.out.println("Execution time in milliseconds: " + timeElapsed / 1000000);
	}
}
```