```
import java.util.Hashtable;
import java.util.Random;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import java.util.zip.CRC32;
import java.util.zip.Checksum;

import static java.lang.Math.abs;

public class CRC32main {
    public static Random randomNum = new Random();
    static int collisionCount = 0;

    public static void main (String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter DNA Length(n): ");
        int dnaLength = input.nextInt();
        System.out.print("Enter Substring Lengths(k): ");
        int subStringLength = input.nextInt();
        input.close();

        int subStringLoopCount = dnaLength - subStringLength + 1;

        //Declare hashtable and its size to the length of the dna (n)
        Hashtable<Integer, Substring> hashTable = new Hashtable<>(dnaLength);

        String DNAString = StringGenerator(randomNum, dnaLength, subStringLength,
subStringLoopCount);

        long start = System.nanoTime();
        //Calls the addToHashTable() method to add the dna substrings to the
hashtable
        addToHashTable(hashTable, DNAString, dnaLength, subStringLength,
subStringLoopCount);
        //Prints table and total number of collisions
        for(int i = 0; i < dnaLength; i++){
            if (hashTable.containsKey(i) == true)
                System.out.println("Key "+ i + ":" +hashTable.get(i).string + "-" +
hashTable.get(i).occurrences);
        }
        System.out.println("Collision count: " + collisionCount);
        long end = System.nanoTime();
        long convert = TimeUnit.MILLISECONDS.convert(end-start,
TimeUnit.NANOSECONDS);
        System.out.print("\nTime in miliseconds to solve K-mer distribution: ");
        System.out.println(convert);
    }

    //String generator
    public static String StringGenerator(Random rand, int nLength, int
subStringLength, int subStringLoopCount) {

        int num;
        int j;
        String stringInput = "";

        for(j = 0; j < nLength; j++) {
            num = rand.nextInt(4);
```

```
            if (num == 0)
                stringInput += 'a';

            if (num == 1)
                stringInput += 'c';

            if (num == 2)
                stringInput += 'g';

            if (num == 3)
                stringInput += 't';
        }
        System.out.println("The DNA Sequence: " + stringInput);

        return stringInput;
    }

    //Hash function crc32
    public static long crc32(String input) {
        byte[] bytes = input.getBytes();
        Checksum checksum = new CRC32(); // java.util.zip.CRC32
        checksum.update(bytes, 0, bytes.length);

        return checksum.getValue();
    }

    //Method that adds the k-mer substrings to hashtable
    static void addToHashTable(Hashtable<Integer, Substring> hashTable, String
DNAString, int hashTableSize, int subStringLength, int subStringLoopCount){
        for(int i = 0; i < subStringLoopCount; i++){
            String temp = DNAString.substring(i,i+subStringLength);
            long result = abs(crc32(temp));
            int key = (int)(result % hashTableSize);
            if(hashTable.containsKey(key) == false){
                hashTable.put(key, new Substring(temp));
            }
            else if (hashTable.containsKey(key) == true){
                if(hashTable.get(key).string.compareTo(temp) == 0)
                    hashTable.get(key).occurrences += 1;
                else{
                    collision(hashTable,key,temp,hashTableSize);
                }
            }
        }

    }
    //Method that checks for collision
    public static boolean collision(Hashtable<Integer, Substring> hashTable, int
key, String temp, int hashTableSize){
        for(int j = 0; hashTable.containsKey(key) == true; j++){
            key = (key + j) % hashTableSize;
            collisionCount++;
            if(hashTable.containsKey(key) == true &&
hashTable.get(key).string.compareTo(temp) == 0){
                hashTable.get(key).occurrences += 1;
                return true;
            }
        }
        hashTable.put(key, new Substring(temp));
```

```
        return true;
    }
}
```