```
/**
*
* author Michael Schukat
*/
public class CT255_HashFunction1 {

   public static void main(String[] args) {
      int res = 0;
      long res2 = 0;
      int coll = 0;
      int i = 0;

      if (args != null && args.length > 0) { // Check for <input> value
         //res = hashF1(args[0]); // call hash function with <input>
         res2 = hashF2(args[0]);
         if (res < 0) { // Error
            System.out.println("Error: <input> must be 1 to 64 characters long.");
         }
         else {
            System.out.println("input = " + args[0] + " : Hash = " + res2);
            System.out.println("Start searching for collisions");

            // Your code to look for a hash collision starts here!
            while (coll < 10) {
               int comp = hashF1(Integer.toString(i));

               //Print result if collision is found
               if (res == comp) {
                  System.out.println("Hash Collision Found: " + i);
                  coll++;
               }
               i++;
            }
         }
      }
      else { // No <input>
         System.out.println("Use: CT255_HashFunction1 <Input>");
      }
   }

   private static int hashF1(String s) {
      int ret = -1, i;
      int[] hashA = new int[]{1, 1, 1, 1};

      String filler, sIn;
```

```
      filler = new
String("ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEF
GH");

      if ((s.length() > 64) || (s.length() < 1)) { // String does not have required length
         ret = -1;
      }
      else {
         sIn = s + filler; // Add characters, now have "<input>HABCDEF..."
         sIn = sIn.substring(0, 64); // // Limit string to first 64 characters
         // System.out.println(sIn); // FYI
         for (i = 0; i < sIn.length(); i++){
            char byPos = sIn.charAt(i); // get i'th character
            hashA[0] += (byPos * 17); // Note: A += B means A = A + B
            hashA[1] += (byPos * 31);
            hashA[2] += (byPos * 101);
            hashA[3] += (byPos * 79);
         }
      hashA[0] %= 255; // % is the modulus operation, i.e. division with rest
      hashA[1] %= 255;
      hashA[2] %= 255;
      hashA[3] %= 255;
      ret = hashA[0] + (hashA[1] * 256) + (hashA[2] * 256 * 256) + (hashA[3] * 256 * 256 *
256);
      }
      if (ret < 0) {
         ret *= -1;
      }
      return ret;
   }

   private static long hashF2(String s) {
      long ret = -1;
      int i;
      long[] hashA = new long[]{1, 1, 1, 1};

      String filler, sIn;

      filler = new
String("ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEF
GH");

      if ((s.length() > 64) || (s.length() < 1)) { // String does not have required length
         ret = -1;
      }
      else {
         sIn = s + filler; // Add characters, now have "<input>HABCDEF..."
```

```
        sIn = sIn.substring(0, 64); // // Limit string to first 64 characters
        // System.out.println(sIn); // FYI
        for (i = 0; i < sIn.length(); i++){
           char byPos = sIn.charAt(i); // get i'th character
           hashA[0] += (byPos * 1543);  // Replaced 17 with 1543
           hashA[1] += (byPos * 3089);  // Replaced 31 with 3089
           hashA[2] += (byPos * 8219);  // Replaced 101 with 8219
           hashA[3] += (byPos * 7109);  // Replaced 79 with 7109
        }
      hashA[0] %= 255; // % is the modulus operation, i.e. division with rest
      hashA[1] %= 255;
      hashA[2] %= 255;
      hashA[3] %= 255;
      //Changed to be multiplied by 2^16
      ret = hashA[0] + (hashA[1] * 65536) + (hashA[2] * 65536 * 65536) + (hashA[3] * 65536 *
65536 * 65536);
      }
      if (ret < 0) {
         ret *= -1;
      }
      return ret;
   }
}
```