CT255 Assignment 2

Part 2:

```
import java.security.SecureRandom;

public class CT255_HashFunction1 {

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

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

					// All the characters that the password generator is able to use.
					String ALPHA_CAPS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
					String ALPHA = "abcdefghijklmnopqrstuvwxyz";
					String NUMERIC = "0123456789";

					int i = 0;

					while (i < 10) {
					String passwd = randomPasswd(10, ALPHA_CAPS + ALPHA + NUMERIC); //
			Calling randomPasswd, letting it create a password 10 characters long with all the above
			characters in the variables
					int tempHash = hashF1(passwd); // Hashing the passwd so we can compare it to
			the hash of the input
					if (tempHash == res) { // Checking to see if the hash is the same
						System.out.printf("Collision found with Password: %s\n", passwd); //
			Printing that a collision has been found
					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("ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGH");

		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[1] += (byPos * 17);
				hashA[2] += (byPos * 31);
				hashA[3] += (byPos * 101);
				hashA[4] += (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 String randomPasswd(int len, String dic) { // Random Password Generator, Takes the length of
password you want to create and a string of characters you want to use
		SecureRandom random = new SecureRandom();

		String result = ""; // Initialising Variable
		for (int i = 0; i < len; i++) { // Repeating length len times
			int index = random.nextInt(dic.length()); // Picking a random number between the range bounds of
the length of the string given.
			result += dic.charAt(index); // Adding the character at position of index in the string dic to result.
		}
		return result; // Returning result back to String passwd
	}
}
```

Part 3:

```
import java.security.SecureRandom;

public class CT255_HashFunction1 {

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

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

					// All the characters that the password generator is able to use.
					String ALPHA_CAPS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
					String ALPHA = "abcdefghijklmnopqrstuvwxyz";
					String NUMERIC = "0123456789";

					int i = 0;

					while (i < 10) {
					String passwd = randomPasswd(10, ALPHA_CAPS + ALPHA + NUMERIC); //
			Calling randomPasswd, letting it create a password 10 characters long with all the above
			characters in the variables
					int tempHash = hashF1(passwd); // Hashing the passwd so we can compare it to
			the hash of the input
					if (tempHash == res) { // Checking to see if the hash is the same
						System.out.printf("Collision found with Password: %s\n", passwd); //
			Printing that a collision has been found
					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("ABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGHABCDEFGH");

		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[i % 4] += (byPos * 17); // Making it random so that there are less collisions
					hashA[(i+1) % 4] += (byPos * 31);
					hashA[(i+2) % 4] += (byPos * 101);
					hashA[(i+3) % 4] += (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 String randomPasswd(int len, String dic) { // Random Password Generator, Takes the length of
password you want to create and a string of characters you want to use
		SecureRandom random = new SecureRandom();

		String result = ""; // Initialising Variable
		for (int i = 0; i < len; i++) { // Repeating length len times
			int index = random.nextInt(dic.length()); // Picking a random number between the range bounds of
the length of the string given.
			result += dic.charAt(index); // Adding the character at position of index in the string dic to result.
		}
		return result; // Returning result back to String passwd
	}
}
```