**UCA spec bugs**

Author: Markus Scherer  
Date: 2012-mar-15

**3.7 Well-Formed Collation Element Tables**

Point 2 says  

All Level N weights in Level N-1 ignorable must be strictly less than all weights in Level N-2 ignorable.

For example, secondaries in non-ignorable must be strictly less than those in primary ignorable: Given collation elements [C, D, E] and [0, A, B], where C ≠ 0 and A ≠ 0, D must be less than A.

This is a contradiction. For N=2, the condition statement says "secondaries weights in primary ignorable must be strictly less than all weights in non-ignorable" which is wrong and contradicts the example.

The correct statement is "All Level N weights in Level N-2 ignorable must be strictly less than all weights in Level N-1 ignorable."

Note from Ken Whistler: This contradiction has been an error all the way back to its introduction in Version 9 of UCA (= UCD 3.1.0 with Corrigendum 3) nearly 10 years ago.
6.10.1 Collation Element Format & 6.10.2 Sample Code

a)
The CE layout has
- expansionsOffset
  - 12 bits = FFF
  - 20 bits = offset (allows for 1,048,576 items)
- contractionsOffset
  - 12 bits = FFE
  - 20 bits = offset (allows for 1,048,576 items)

but the sample code has
```c
void processCE(int ce) {
    if (ce < 0xFFF00000) {
        output[outputPos++] = ce;
    } else if (ce >= 0xFFF00000) {
        copyExpansions(ce & 0x7FFFFFF);
    } else {
        searchContractions(ce & 0x7FFFFFF);
    }
}
```
which neither matches, nor works at all.

For the code to match the CE bits, it would have to be
```c
void processCE(int ce) {
    if (ce < 0xFFE00000) {
        output[outputPos++] = ce;
    } else if (ce >= 0xFFE00000) {
        copyExpansions(ce & 0xFFFFFFF);
    } else {
        searchContractions(ce & 0xFFFFFFF);
    }
}
```
b)

void searchContractions(int offset)
  ● Does not handle discontiguous contractions. That should at least be noted.
  ● Skips or reads the backwardsOffset from input rather than from contractionMatches.
  ● Reads the length entry again as a cc character rather than skipping it.
  ● Has some code to do backwards matching but it only ever reads input[inputPos++].
    Given that none of the rest of the sample code is prepared to work backwards, I suggest
    removing all mentioning of the "forwards" flag and the getCollationElementStart() function. (That one also does not take into account combining marks that might be
    skipped.)
  ● Compares input chars and contraction chars as (short) which is a signed type. The
    test "cc > goal" will fail if input is ≥ U+8000.

An improved version of searchContractions() might look like this:
void searchContractions(int offset) {
  offset++;  // skip backwardsOffset
  int goal = input[inputPos++];
  int length = contractionMatches[offset];
  int limit = offset + 1 + length;
  for (int i = offset + 1; i < limit; ++i) {
    int cc = contractionMatches[i];
    if (cc > goal) { // definitely failed
      processCE(contractionCES[offset]);
      break;
    } else if (cc == goal) { // found match
      processCE(contractionCES[i]);
      break;
    }
  }
}


c)

The sample code does not work for supplementary code points, but I guess that is meant as an
exercise for the reader. It should be noted as such.