ISO/IEC JTC1/SC 22/WG 20 N 731

Date: 1999-12-23

ISO
ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
□□

CEI (IEC)
COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE
INTERNATIONAL ELECTROTECHNICAL COMMISSION
□□

Title	**ISO/IEC DIS 14651** - International String Ordering - Method for comparing Character Strings and Description of the Common Template Tailorable Ordering *[ISO/CEI DIS 14651 - Classement international de chaînes de caractères - Méthode de comparaison de chaînes de caractères et description du modèle commun d'ordre de classement]*
Status:	Final Draft International Standard **(**D**R**A**F**T** **D**I**S**)**
Reference:	SC22/WG20 N 670 (Disposition of comments on FCD 14651.2)
Date:	1999-12-23
Project:	22.30.02.02
Editor:	Alain LaBonté Gouvernement du Québec Secrétariat du Conseil du trésor 875, Grande Allée Est, Section 3C Québec, QC G1R 5R8 Canada
Email:	alb@sct.gouv.qc.ca

Contents:

FOREWORD

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee known as ISO/IEC JTC1. Draft International Standards adopted by the joint technical committee are circulated to the national bodies for voting. Publication as an international standard requires approval by at least 75% of the national bodies that cast a vote.

The ISO/IEC 14651 International Standard has been prepared by the Joint Technical Committee ISO/IEC JTC1, Information Technology.

INTRODUCTION

This International Standard provides a method for ordering text data world-wide, and provides a Common Template Table whose tailoring meets the requirements of given languages and cultures while retaining reasonable ordering properties for other scripts.

The Common Template Table requires some tailoring in different local environments. However conformance to this International Standard requires that all deviations from the Template, called "deltas", be declared to document resultant discrepancies.

This International Standard describes a method to order text data independently of context.

1 Scope

This International Standard defines:

- A reference comparison method applicable to two strings in order to determine their respective order in a sorted list. The method can be applied to strings containing characters from the full repertoire of ISO/IEC 10646-1. This method is also applicable to subsets of that repertoire, such as those of the different ISO/IEC 8-bit standard character sets, or any other character set, standardized or private, to produce ordering results valid (after tailoring) for a given set of languages for each script. This method uses collation tables derived either from the Common Template Table defined in this International Standard or from one of its tailorings.

- A reference format, using the Backus-Naur Form (BNF) to describe the Common Template Table used normatively within this International Standard.

- A Common Template Table used by the reference comparison method. This table describes an order for all characters encoded in the first edition of ISO/IEC 10646-1 up to Amendment 7. It allows for a specification of a fully deterministic ordering. This table enables the specification of an international string ordering adapted to different cultures, without requiring an implementor to have a knowledge of all the different scripts already encoded in the UCS.

NOTE 1: This Common Template Table is to be modified to suit the needs of a local environment. The main benefit, worldwide, is that for other scripts, often no modification may be required and that the order will remain as consistent as possible and predictable from an international point of view.

NOTE 2: The character repertoire used in this International Standard is equivalent to that of the Unicode Standard Version 2.1.

- A reference name, representing this particular version of the Common Template Table for use as a reference for tailoring. In particular, this name implies that the table is linked to a particular stage of development of the ISO/IEC 10646 Universal multiple-octet coded character set.

- Requirements for a declaration of the differences (delta) between the collation table and the Common Template Table.

This International Standard does *not* mandate:

- A specific comparison method; any equivalent method giving the same results is acceptable.
- A specific format for describing or tailoring tables in a given implementation.
- Specific symbols to be used by implementations except the name of the Common Template Table.
- A specific user interface for choosing options.
- A specific internal format for intermediate keys used in comparisons nor for the table used. The use of numeric keys is not mandated either.
- A context-dependent ordering.

- Any particular preparation of character strings prior to comparison.

NOTE 1: It is normally necessary to do preparation of character strings prior to comparison even if it is not prescribed by this International standard (see informative annex C).

NOTE 2: Although no user interface is required to choose options or to specify tailoring of the Common Template Table, conformance requires always declaring the applicable delta, a declaration of differences with this table. It is recommended that processes present available tailoring options to users.

2 Conformance

A process is conformant to this International Standard if it meets the requirements prescribed in subclauses 6.2 to 6.5.

A declaration of conformity to this International Standard shall be accompanied by a statement, either directly or by reference, of the following:

- the number of levels that the process supports. This number shall be at least three;

- whether the process supports the `position` parameter;

- whether the process supports the `backward` parameter and at which level;

-the tailoring *delta* described in 6.4 and how many levels are defined in the delta.

It is the responsibility of implementors to show how their delta declaration is related to the table syntax described in 6.3, and how the comparison method they use, if different from the one mentioned in clause 6, can be considered as giving the same results as those prescribed by the method specified in clause 6.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Members of IEC and ISO maintain registers of currently valid International Standards.

- ISO/IEC 10646-1:1993 *Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane*

- ISO/IEC 10646-1:1993/Amd.1:1996 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 1: Transformation Format for 16 planes of group 00 (UTF-16)*

- ISO/IEC 10646-1:1993/Amd.2:1996 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 2: UCS Transformation Format 8 (UTF-8)*

- ISO/IEC 10646-1:1993/Amd.4:1996 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 4*

- ISO/IEC 10646-1:1993/Amd.5:1998 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 5: Hangul syllables*

- ISO/IEC 10646-1:1993/Amd.6:1997 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 6: Tibetan*

- ISO/IEC 10646-1:1993/Amd.7:1997 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 7: 33 additional characters*

- ISO/IEC 10646-1:1993/Amd.9:1997 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 9: Identifiers for characters*

- ISO/IEC 10646-1:1993/Amd.18:1997 *Information technology – Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane Amendment 18: Symbols and other characters*

NOTE concerning Amendment 18: Only the EURO SIGN and the OBJECT REPLACEMENT CHARACTER from Amendment 18 are accounted for in Annex A of this International Standard at this time.

4 Definitions

For the purposes of this International Standard, the following definitions apply:

4.1 character string	a sequence of characters**4.2 collation** equivalent to the term “ordering”
4.3 collating symbol	a symbol used to specify weights assigned to a collating element
4.4 collation (weighting) table	a mapping from collating elements to weighting elements
4.5 collating element	a sequence of one or more characters that are considered a single element for ordering
4.6 delta	differences from a given collation table.The given collation table, together with a given delta, forms a new collation table. Unless otherwise specified in this International Standard, the term "delta" always refers to differences from the Common Template Table as defined in this International Standard
4.7 (collation) level	the sequence number for a subkey
4.8 ordering	a process by which two strings are determined to be in exactly one of the relationships of **less than**, **greater than**, or **equal** to one another
4.9 ordering key	a sequence of subkeys used to determine an order
4.10 (collation) preparation	a process in which given character strings are mapped to (other) character strings logically before the calculation of the ordering key for each of the strings
4.11 reference comparison method	the method for establishing an order between two ordering keys (see clause 6)
4.12 subkey	a sequence of weights computed for a character string for a particular level.
4.13 symbol	collating element
4.14 (collation) weight	a positive integer value, used in subkeys, reflecting the relative order of collating elements
4.15 weighting element	a list of a given number of weights sequentially ordered by level

5 Symbols and abbreviations

Following ISO/IEC_10646-1 requirement (Amendment 9) characters are referenced as UXXXX where X stands for any hexadecimal digit (using upper case letters where applicable) and refers to the value of that character in ISO/IEC 10646-1. This convention is used throughout this International Standard.

In the Common Template Table arbitrary symbols representing weights are used according to the BNF notation description in *6.3.1*.

6 String comparison

6.1 Preparation of character strings prior to comparison

It may be necessary to transform character strings before the reference comparison method is applied to them (see annex C for an example of such preparation). Although not part of the scope of this International Standard, preparation may be an important part of the ordering process, as for example in telephone-book ordering, a complex case in point.

Where applicable, it can be an important part of the prehandling phase to map characters from a non-UCS encoding scheme to the UCS for input into the reference comparison method. This task can amongst other things encompass the correct handling of escape sequences in the originating encoding scheme, the mapping of characters without an allocated UCS codepoint to an application-defined codepoint in the private zone area and inverting strings which are not stored in logical order. For example, visual order Arabic code sets must be put into logical order; bibliographic code sets with accents before base characters require reversal. The resulting string sequence may then have to be remapped into its original encoding scheme.

NOTE 1: The Common Template Table is designed so that combining sequences and corresponding single characters (precomposed) will have precisely the same ordering. To avoid inadvertently breaking this invariant (and in the process breaking Unicode conformance), tailoring should reorder combining sequences when corresponding characters are reordered. For example, if Ä is reordered after Z, then the sequence <A>+<combining diaeresis> should also be reordered. To avoid exposing encoding differences that may be invisible to the end-user, it is recommended that strings be normalized according to Unicode normalization to achieve this equivalence – see Bibliography, Unicode Technical Report no. 15.

NOTE 2: Escape sequences and control characters constitute very sensitive data to interpret, and it is highly recommended that preparation should filter out or transform these sequences.

NOTE 3: Since the reference method is a logical statement for the mechanism for string comparison, it does not preclude an implementation from using a non-UCS character encoding only, as long as it produces results as if it were using the reference comparison method.

6.2 Key building and comparison

6.2.1 Preliminary considerations

6.2.1.1 Assumptions

The collation table is a mapping from collating elements to weighting elements. In each weighting element, four levels are described in the Common Template Table. This number of levels can be extended or reduced (but not below 3 levels) in tailoring.

NOTE: Normally the last level is intended to specify "special" characters, i.e., characters normally not part of the spelling of words of a language (such as dingbats, punctuation, etc.), sometimes called "ignorable" characters in the context of computerized ordering.

6.2.1.2 Processing properties

The whole table has specific scanning and ordering properties. These properties may be changed.

The scanning direction (forward or backward) used to process the string is a global property of each level defined in the table.

An optional property of the last level of comparison is that, before comparing weights of each "ignorable" character, a comparison on the numeric position of each such character in the two strings may be effected. This property is known as the position parameter. In other words, for two strings equivalent at all levels except the last one, the string having an ignorable in the lowest position comes before the other one. In case ignorables share the same positions, then weights are considered, until a difference is found). This processing is optional and is not necessary to claim conformance.

NOTE: *The scanning direction (forward or backward) is not normally related to the natural writing direction of scripts. The scanning direction applies to the logical sequence of the coded character string.*

According to ISO/IEC 10646, for scripts written right to left, such as Arabic, the lowest positions in the logical sequence of characters correspond to the rightmost characters of a string (from the point of view of their natural presentation sequence). Conversely, for the Latin script, written left to right, the lowest positions in the logical sequence of characters correspond to the leftmost characters of the string (from the point of view of their natural presentation sequence).

Scanning forward starts with the lowest position in the logical sequence, while scanning backward starts from the highest position, independently of the presentation sequence. The scanning direction for ordering purposes is a global property of each level described in the table.

In ISO/IEC 10646-1, the Arabic script is artificially separated into two pseudoscripts: 1) the logical, intrinsic Arabic, coded independently of shapes, and 2) the Arabic presentation forms. Both allow the complete coding of Arabic, but intrinsic Arabic is normally preferred for better processing, while presentation-form Arabic is preferred by some presentation-oriented applications. ISO/IEC 10646-1 does not prescribe that the presentation forms be coded in any specific order and in some implementations, the coded order for the latter is exactly the opposite of the one used for “intrinsic Arabic”. It is therefore eventually advisable that prehandling be used to make sure that coded Arabic presentation forms and other coded Arabic characters be fed to the comparison method in the same logical order.

A tailored table may be separated into sections for ease of tailoring. Each section is then assigned a name consistent with the specification in subclause 6.3.1. One of the tailoring possibilities is to assign a given order to each section and to change the relative order of an entire section relative to other sections.

6.2.2 Ordering key formation

When two strings are compared to determine their relative order, the two strings are first broken up into a sequence of collating elements taking account of multi-character collating elements using "`collating_element`" statement in a tailored table. Then a sequence of *m* intermediary subkeys is formed out of a character string, where *m* is the number of levels described in a tailored collation weighting table. The following subclauses describe the formation of an ordering key.

Each ordering key is a sequence of subkeys. Each subkey is a list of numeric weights formed by successively appending each of the weights assigned to corresponding characters at a given level. There are three types of subkeys, subkeys using the "`forward`" parameter, subkeys using the `backward` parameter and subkeys using the `forward position` parameter. Subkeys that use the position parameter may only occur at the last level. Support of the position parameter is not required for conformance but shall be declared if used.

The special keyword "`IGNORE`" as a weight indicates that when strings are compared using the weights at the level where "`IGNORE`" is specified, the collating element shall be ignored; i.e., as if the string did not contain the collating element.

If there are no weights corresponding to a character of the input string, then the character is *undefined*. Undefined characters are ordered with respect to defined characters as if they were given the weight “`UNDEFINED`“ at any particular level in the Common Template Table. If there is no weight assigned to symbol “`UNDEFINED`“ in the table, then the table is interpreted as if there were one at the very end. The ordering of undefined characters with respect to other undefined characters is not specified in this standard.

NOTE: *there are two common treatments of* undefined *characters. The first is to sort among them as if their level-one weight differences were based upon their UCS character code. The second is to sort them as if they all* *had the same level-one weight, and their second-level weights were the same as their UCS character codes.*

6.2.2.1 Formation of a subkey with the `forward` parameter

Subkeys with the `forward` parameter are formed in the following way at a particular level:

During forward scanning of each character of the input character string, one or more weights are obtained. These weights are obtained by matching the UCS value of each character, expressed as a hexadecimal string, against a corresponding symbol prefixed with “U” in a tailored collation weighting table, and then evaluating the weight or weights associated with that level.These weights are appended to the end of the subkey.

6.2.2.2 Formation of a subkey with the `backward` parameter

Subkeys with the `backward` parameter are formed by forming a subkey with the `forward` parameter at a particular level, then reversing that subkey.

6.2.2.3 Formation of a subkey which uses the `forward position` parameter

Subkeys using the `forward position` parameter are formed at a particular level in the following way:

During forward scanning of each collating element of the input character string, two weights are appended to the end of the subkey. The first weight corresponds to the logical position in the original character string of the collating element being processed. The second weight corresponds to the weight assigned to that collating element at the subkey’s level in the table.

NOTE: In the Common Template Table, levels generally have the following characteristics, although the purpose of each level is not absolute:

Level 1: This level generally corresponds to the set of common letters of the alphabets for that script, if the script is alphabetic, and to the set of common characters of the script if the script is ideographic or syllabic.

Level 2: This level generally corresponds to diacritical marks affecting each basic character of the script. For some languages, letters with diacritics are always considered an integral part of the basic letters of the alphabet, and are not considered at this second level, but rather at the first. For example, in Spanish, N TILDE is considered a basic letter of the Latin script. Therefore, tailoring for Spanish will change the definition of N TILDE from "the weight of an N in the first level and the weight of a TILDE in the second level" to "the weight of an N TILDE (placed after N and before O) in the first level, and indication of the absence of a diacritic in the second level". For some characters, variant letter shapes are also dealt with on level 2. An example of this is ß, the LATIN LETTER SHARP S, which is treated as equivalent to ss on level 1, but traditionally distinguished from it on level 2.

Level 3: This level generally corresponds to case distinctions or, mainly in non-Latin scripts, to distinctions based on variant character shapes.

Level 4: This level generally corresponds to weighting differences that are less significant than those at the other levels.

6.2.3 Reference comparison method for ordering character strings

The following describes the reference comparison method:

1. In considering a table describing weights (each instance **a** of weight is noted **wt(a)** below) at *m* levels (an instance of a subkey at a given level **b** is noted **sk(b)** below) for each of *n* characters in the implementation character set, build an ordering key for each of two arbitrary character strings

being compared, according to the algorithm of key formation described in sub-clause 6.2.2of this International Standard.

2. An ordering key **x** is less than an ordering key **y** at level **s** where $\mathbf{1PsPn}$ if and only if there exists some integer **i** where $\mathbf{1PiPs}$ such that $\mathbf{x_{sk(i)}} < \mathbf{y_{sk(i)}}$ and $\mathbf{x_{sk(i)}} = \mathbf{y_{sk(i)}}$ for all integers **j** where $\mathbf{1PjPi}$.

   An ordering key **x** is greater than an ordering key **y** if and only if **y** is less than **x**;

   An ordering key **x** is equal to an ordering key **y** if neither **x** is less than **y** nor **y** is less than **x.**

3. A subkey **v** of length $\mathbf{l_v}$ is less than a subkey **w** of length $\mathbf{l_w}$ if and only if $\mathbf{l_v=0}$ and $\mathbf{l_w>0}$ or if there exists some integer **p** where $\mathbf{pPl_v}$ and $\mathbf{pP\ l_w}$ such that for all integers **q**, where $\mathbf{1PqPp}$, $\mathbf{v_{wt(q)}=w_{wt(q)}}$ and either $\mathbf{l_v=p}$ and $\mathbf{l_w\Sigma p}$ and $\mathbf{v_{wt(p)}=w_{wt(p)}}$ or $\mathbf{v_{wt(p)}<w_{wt(p)}}$.

   A subkey **v** is greater than a subkey **w** if and only if **w** is less than **v** ;

   A subkey **v** is equal to a subkey **w** if neither **v** is less than **w** nor **w** is less than **v.**

The table used in this reference comparison method is the result of the numeric interpretation of the symbols in a tailored table.

6.3 Common Template Table: formation and interpretation

This clause specifies:

- the syntax used to form the Common Template Table in annex A of this International Standard or a tailored table based upon the Common Template Table
- conditions of well-formedness of a table using this syntax
- interpretation of tables formed using this syntax
- conditions for considering two tables as equivalent
- conditions for considering comparison results as equivalent

6.3.1 BNF syntax rules

Definitions between <angle brackets> make use of terms not defined in this BNF syntax, and assume general English usage.

Other conventions:
* indicates 0 or more repetitions of a token or a group of tokens
+ indicates 1 or more repetitions of a token or a group of tokens
? indicates optional occurrence of a token or a group of tokens (0 or 1 occurrences)
parentheses are used to group tokens
production rules are terminated by a semicolon
comments start with a % character on a separate line

```
% Define collation tables as sequences of lines.
weight_table = common_template_table | tailored_table ;
tailored_table = table_line+ ;
common_template_table = simple_line+ ;

% Define the line types.
table_line = simple_line | tailoring_line ;
tailoring_line = (section_definition | reorder_after | reorder_end |
            reorder_section_after | order_start | order_end)
            line_completion ;
simple_line = (symbol_definition | weight_assignment | collating_element)?
            line_completion ;

% Define the tailoring syntax.
order_start = 'order_start' space+ multiple_level_direction (',position')?
            ;
order_end = 'order_end' ;
multiple_level_direction = direction (semicolon direction)* ;
direction = 'forward' | 'backward' ;
reorder_section_after = 'reorder-section-after' space+ section_identifier
            space+ target_symbol ;
reorder_after = 'reorder-after' space+ target_symbol ;
reorder_end = 'reorder-end' ;
target_symbol = symbol ;
section_definition = section_definition_simple | section_definition_list ;
section_definition_list = 'section' space+ section_identifier space+
            symbol_list ;
section_definition_simple = 'section' space+ section_identifier ;
section_identifier = identifier ;
symbol_list = symbol_element (semicolon symbol_element)* ;

% Define the basic syntax for collation weighting.
collating_element = 'collating-element' space+ symbol space+ 'from' space+
            quoted_symbol_sequence ;
weight_assignment = simple_weight | symbol_weight ;
simple_weight = symbol_element ;
symbol_weight = symbol_element space+ weight_list ;

symbol_definition = ('collating-symbol' space+ symbol_element) |
            'UNDEFINED' ;
weight_list = level_token (semicolon level_token)* ;
level_token = symbol_group | 'IGNORE' ;
symbol_group = symbol_element | quoted_symbol_sequence ;
quoted_symbol_sequence = '"' symbol+ '"' ;
symbol_element = symbol | symbol_range ;
symbol_range = symbol '..' symbol ;
symbol = simple_symbol | ucs_symbol ;
ucs_symbol = ('<U' four_digit_hex_string '>') | ('<U-'
            eight_digit_hex_string '>') ;
simple_symbol = '<' identifier '>' ;

% Define low-level tokens used by the rest of the syntax.
identifier = (letter | digit) id_part* ;
id_part = letter | digit | '-' | '_' ;
line_completion = space* comment? EOL ;
comment = comment_char character* ;
eight_digit_hex_string = hex_upper hex_upper hex_upper hex_upper hex_upper
            hex_upper hex_upper hex_upper ;
four_digit_hex_string = hex_upper hex_upper hex_upper hex_upper ;
hex_numeric_string = hex_upper+ ;
space = ' ' | <TAB> ;
semicolon = ';' ;
comment_char = '%' ;
letter = 'a' | 'b' | 'c' | 'd' |'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' |
            'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v'
            | 'w' | 'x' | 'y' | 'z' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' |
            'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q'
            | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' ;
hex_upper = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | digit ;
digit = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
EOL = <end-of-line in the text conventions in use> ;
character = <any member of the repertoire of the encoded character set in
            use, not including any characters used to delimit the end of
            lines> ;
```

6.3.1.1 Keyword usage

The following keywords were inherited from POSIX locale syntax. The original usage of these keywords is as follows:

collating-element	Define a collating-element symbol representing a multicharacter collating element. This keyword is optional.
order_start	Define collation rules. This statement is followed by one or more collation order statements, assigning character collation values and collation weights to collating elements.
order_end	Specify the end of the collation-order statements.
reorder-after	Redefine collating rules. Specify after which collating element the redefinition of collation order shall take order. This statement is followed by one or more collation order statements, reassigning character collation values and collation weights to collating elements.
reorder-end	Specify the end of the "reorder-after" collating order statements.
reorder-section-after	Redefine the order of sections. This statement is followed by one or more section symbols, reassigning character collation values and collation weights to collating elements.
reorder-section-end	Specify the end of the "reorder-sections" section order statements.

6.3.2 Well-formedness conditions

WF1. Any *simple_symbol* occurring in a *weight_list* shall occur in a *symbol_definition* in the same *symbol_weight* in which the *weight_list* occurs, or in a *symbol_weight* or *symbol_definition* that occurs earlier in the sequence of lines that constitute a *weight_table*.

NOTE: All simple_symbol*s shall be "defined" before they are "used".*

WF2. No *symbol* that occurs in a *symbol_definition* in a *weight_table* that contains no *tailoring_line*'s may occur in another *symbol_definition* in the same *weight_table*.

NOTE: Duplication of collation weighting symbols is prohibited. This is true for the Common Template Table itself, and must also remain true for a tailored_table*, once the results of all reordering of lines has been applied.*

WF3. All *weight_list*'s in a *tailored_table* shall contain the same number of *level_token*'s. An empty *level_token* shall be interpreted as the *collating_element* itself.

NOTE: A tailorable table shall be consistent in its use of levels throughout.

WF4. A *tailored_table* shall contain one *order_start* statement..

NOTE: This statement shall appear after symbol_definition entries and before the symbol_weight entries.

WF5. A *multiple_level_direction* in a *tailored_table* shall contain the same number of *direction*'s as the number of *level_token*'s of any *weight_list* in that *tailored_table*.

NOTE: Any order_start *shall have the same number of levels as is generally used in the table.*

WF6. If a *level_token* in a *weight_list* consists of a *symbol_group*, all successive *level_token*'s in that *weight_list* shall also consist of a *symbol_group*.

NOTE: 'IGNORE' shall not be used at a level after an explicit symbol for a weighting.

WF7. Any *section_identifier* occurring in a *reorder_section_after* shall occur in a *section_definition* which occurs earlier in the sequence of *table_line*s that constitutes a *tailored_table*. All *section_identifier*s shall be "defined" before they are "used".

WF8. No two *section_definition_entry*s in a *tailored_table* shall contain the same values in their *section_identifier*s.

NOTE: Multiple definition of section*s is prohibited;* section_identifier*s shall be unique.*

WF9. Each *reorder_after* in a *tailored_table* shall be followed at a later point in that tailored table by a *reorder_end* or another *reorder_after*.

WF10. A *tailored_table* shall contain one *order_start* and one *order_end*.

WF11. No *reorder_section_after* shall contain a *target_symbol* whose value is the same as any *symbol* in the *section_definition_list* whose *section_identifier* is the same as the *section_identifier* in that *reorder_section_after*.

NOTE: A section *shall not be reordered after a line which the* section *itself contains; attempts at recursive relocation of lines are prohibited.*

WF12. Any *symbol_range* shall contain two *symbol*'s which meet the following conditions: Each *symbol* shall contain a common prefix; the portions of *identifier* of each *symbol* following the common prefix shall be a *hex_numeric_string;* and when interpreted as numeric values, the *hex_numeric_string* of the first *symbol* shall be less than the *hex_numeric_string* of the second *symbol*. The positive integral difference between the *hex_numeric_string* of the second *symbol*, interpreted as a numeric value, and the *hex_numeric_string* of the first *symbol*, interpreted as a numeric value, constitutes the range of *values* of the *symbol_range*.

NOTE: A well-formed symbol_range *is of a form such as <S4E00>..<S9FA5>, where the common prefix is "S", and the rest of the* identifier *portion of each* symbol *is a* hex_numeric_string.

WF13. Any *symbol_weight* which contains more than one *symbol_range* shall contain only *symbol_range*'s that meet the following requirement: Each *symbol_range* following the first *symbol_range* shall have the same number of values in the *range* as that of the first *symbol_range*.

NOTE: This condition guarantees that all expanded ranges will be well-formed, since for any one symbol_weight*, all of the range expansions will have the same number of values.*

6.3.3 Interpretation of tailored tables

I1. A *section* consists either 1) of the list of *simple_line*s which contain a *symbol_definition* whose value is equal to any *symbol* contained in the *symbol_list* in a *section_definition_list_entry*, or 2) of the list of *simple_line*'s following a *section_definition_simple* in a *tailored_table*.

NOTE: A section *is defined 1) by a specific* symbol_list*, or 2) by taking all the lines following the* section_definition *until another tailoring line such as an* order_start*, a* reorder_section_after*, another* section_definition*, or the end of the entire table is encountered.*

I2. A *simple_line* consisting of a *symbol_definition* containing a *symbol_range* is equivalent to a sequence of *simple_line*'s, each containing a *symbol* in place of the *symbol_range*, where the *symbol* for each successive *simple_line* is generated by concatenating a *hex_numeric_string* to the common prefix of the *symbol_range,* in numeric order, starting with the hex value associated with the *hex_numeric_string* of the first *symbol*, and ending with the hex value associated with the *hex_numeric_string* of the second *symbol*. The *hex_numeric_string* concaneted to the common prefix must contain the same number of digits as the *hex_numeric_string* of the first symbol. The number of *simple_line*'s thus generated is equal to the number of symbols in the *symbol_range*.

NOTE: A symbol_definition *of the form "collating-symbol <S0301>..<S0303>" is equivalent to the three lines:*
collating-symbol <S0301>
collating-symbol <S0302>
collating-symbol <S0303>

I3. A *simple_line* consisting of a *symbol_weight* containing one or more *symbol_range*'s is equivalent to a sequence of *simple_line*'s, where each *symbol_range* has been expanded into a sequence of *symbol*'s, as described in I2 for *symbol_definition*'s.

NOTE: A symbol_weight *of the form "<U2000>..<U2002> <S0301>..<S0303>;<BLANK>;<MIN>; <U2000>..<U2002>" is equivalent to the three lines:*
<U2000> <S0301>;<BLANK>;<MIN>;<U2000>
<U2001> <S0302>;<BLANK>;<MIN>;<U2001>
<U2002> <S0303>;<BLANK>;<MIN>;<U2002>

I4a. A *tailored_table* containing a *reorder_after* is equivalent to the *tailored_table where*:

1) have been removed all *table_line*'s that were ahead of the *reorder_after and* that contained *symbol_definition*'s whose *symbol* matches the *symbol* of any *symbol_definition* in the *table_line*'s between the *reorder_after* and *reorder_end*,
2) have been reordered the *table_line*s between that *reorder_after* and the first subsequent *reorder_end* to immediately follow the first *table_line* in the *tailored_table* containing a *symbol_definition* whose *symbol* is the same as the *target_symbol* in the *reorder_after*, and
3) have been removed that *reorder_after* and that *reorder_end*.

NOTE: Move the block of lines between the reorder_after *and the* reorder_end *to follow the* target_symbol*, delete any prior lines that duplicate the* symbol_definition*'s of the reordered lines, and remove the* reorder_after *and* reorder_end *themselves.*

I4b.: When a *tailored_table* contains multiple groups of lines to be reordered, the table is interpreted by processing each *reorder_after* sequentially, starting from the first line of the table.

NOTE: Subsequent line reorderings may impact lines that themselves were reordered by prior reorderings.

I5. A *tailored_table* containing a *section_reorder_after* is equivalent to the *tailored_table* with the *section* associated with that *section-reorder_after* reordered (in the same relative order as the *table_line*s have in that *section*) to immediately follow the last *table_line* in the *tailored_table* containing a *symbol_definition* whose *symbol* is the same as the *target_symbol* in the *section_reorder_after*, and with that *section_reorder_after* removed.

I6. A *weight_table* is said to be in normal form when it contains no *reorder_after*'s or *section_reorder_after*'s.

NOTE: A tailored_table *can be put into normal form by the operations implied by I4 and I5.*

6.3.4 Evaluation of weight tables

E1. A *weight_table* in normal form is said to be evaluated when each *weight_assignment* in the *weight_table* is mapped to a positive integer value (a weight) such that those values increase monotonically by the order in which the *weight_assignment*'s occur in the *weight_table*.

NOTE 1:The table_line'*s of the* weight_table *are first mapped to the set of positive integers, by sequential order in the table. This mapping defines an ordered set of line numbers. The* weight_assignment'*s are then mapped to a set of positive integers (weights) that varies monotonically with the set of line numbers.*

NOTE 2: this does not restrict the starting number for the weight of the first weight_assignment*, nor does it require that the numbers for these weights be immediately consecutive.*

E2. An evaluated *weight_table* is said to be collation-element-weighted when each *simple_symbol* occurring in each *weight_list* in that evaluated *weight_table* has been mapped to the weight which corresponds to the *weight_assignment* which contains the same *simple_symbol*.

NOTE 1: Each weight_list *can be interpreted as containing either* symbol'*s mapped to integral weight values, or as instances of the string 'IGNORE'. At this point the mathematical injection of strings can be defined using the* weight_table*. The string 'IGNORE' is equivalent to an empty list of weight symbols.*

NOTE 2: In a tailored table the value of any hex_numeric_string *associated with a* symbol *typically does not reflect the numeric weighting of the* symbol*.*

6.3.5 Conditions for considering specific table equivalences

A *weight_table* W1 and a *weight_table* W2 are said to be equivalent at a particular level if any comparison of strings using those tables up to that level results in the same ordering.

NOTE: If one takes two strings, builds keys for each based on W1 and compares them, one should always get the same results as when one builds keys for those strings based on W2 and compares them.

6.3.6 Conditions for results to be considered equivalent

An implementation of international string ordering is conformant with this International Standard if for any set of strings S defined on a repertoire R, the implementation can duplicate the same comparisons as those resulting from comparison of the numbers from an injection constructed according to the rules of clause 6.2.3 of this International Standard.

6.4 Declaration of a delta

Tailoring shall be based upon the Common Template Table described in annex A. Tailoring may be accomplished using any syntax that is equivalent to the one described in this International Standard.

NOTE: *For example, ISO/IEC DTR 14652, uses a compatible extension of the syntax used in this International Standard for tailoring A tailoring delta can also be expressed using the syntax of the Unicode collation algorithm (see Bibliography - Unicode Technical Report no. 10). It has also been demonstrated that a tailoring delta can also be expressed using an XML-conformant markup scheme.*

Any declaration of conformance to this International Standard shall be accompanied with a declaration of the differences between the collation weighting table and the Common Template Table. A delta shall contain the equivalent of:

1. At least one valid *order_start* entry described in clause *6.3.1*; an unlimited number of sections containing an *order_start* entry and an *order_end* entry may be declared.

2. The number of levels used for comparison.

3. The list of *symbol_definition* weights (as defined in 6.2.1) added and after which *symbol_definition* entry each insertion is made.

4. The list of *simple_line* entries (as defined in 6.2.1) deleted or inserted, referencing after which *simple_line* entry in the Common Template Table the insertions are made

NOTE: It is recommended that a delta should not be bigger than necessary.

In cases where a process has provision to allow the end-user to tailor the table himself or herself, a statement of conformance shall indicate which of the 4 elements of the previous list are tailorable and which of those 4 elements are not tailorable. For those which are not tailorable, the delta of fixed elements relative to the Common Template Table shall be declared.

NOTE: *The declaration may use a different syntax from the one specified in 6.3 provided that the relationship with this syntax can be reasonably established. For example, the following declarations are valid:*

"Collate U+00E5 after U+007A at the primary level.
Collate U+00E4 after U+00E5 at the primary level. "

or

"The primary alphabet order is modified so that in all cases z < å < ä"

These two notations can reasonably be considered to be equivalent to the more precise expressions (which also give weights at levels 2 and 3):

******************** THOSE PARAGRAPHS IN PINK TO BE MODIFIED BY KEN *******************

```
reorder-after <S007A>
<U00E5> <U00E5>;<BLANK>;<MIN>
reorder-end

reorder-after <S00E5>
<U00E4> <U00E4>;<BLANK>;<MIN>
reorder-end
```

6.5 Name of the Common Template Table and name declaration

Whenever the Common Template Table is referred externally as a base point in a given context, whether in a process, contract, or procurement requirement, it shall be referenced using the name

ISO14651_1999_TABLE1. If another name is used due to practical constraints, a declaration of conformance shall indicate how the correspondence between this other name and the name ISO14651_1999_TABLE1 is taken care of.

The use of a defined name is necessary to manage the different stages of development of this table. This follows from the nature of the reference character repertoire, for which development will be ongoing for a number of years or even decades.

Annex A -- Common Template Table (normative)

In this ordering table, a number of characters and scripts of the world are missing, due to the fact that those characters or scripts have not yet been encoded in the reference character set repertoire, ISO/IEC 10646-1 (Universal multiple-octet coded Character Set, or UCS) at time of the preparation of this International Standard.

It is the intent of ISO/IEC to include the ordering of those scripts explicitly in the Common Template Table when data becomes available, by way of amendments to this International Standard. If the Common Template Table is not tailored for unspecified characters, then an implicit order is assigned in the following table, which may not meet the user requirements of a particular community. Any delta with this table shall be declared in a statement of conformance to this International Standard as per the specifications of the conformance clause.

Name used for referring to this table in this version of this International standard: **ISO14651_1999_TABLE1**

************ TEXT TO BE MODIFIED WHEN WE KNOW THE PERMANENT URL OF THE MACHINE_READABLE TABLE ON ITTF SITE ********

NOTE: Some lines in the following table spill in presentation because of long line lengths. An eventual machine-readable table using this format shall be corrected to avoid this minor presentation problem. In case of doubt a copy of the source file (which does not present this presentation problem as it is a machine-readable file) is available from the SC22/WG20 convenor or from the editor of this International Standard. At time of publishing of the International Standard, it is currently the intent that the machine-readable table will be assigned a permanent URL which will be refered in the final document in addition to being reproduced fully in the International Standard.

```
% escape_char /
□% comment_char %
```

% LC_COLLATE

```
□% Decomment the lines above to create a 14652-style
%   LC_COLLATE definition.


************************ THE FULL TABLE COMES AROUND HERE********************
UNDEFINED % For all unsupported characters to be treated as unweighted.


% order_end

% END LC_COLLATE

% Decomment the line above to create a 14652-style
%   LC_COLLATE definition.
```

Annex B – Tailoring deltas (informative)

B.1 Example 1 – Canadian delta and benchmark

This annex describes benchmark 1, based on Canadian standard CAN/CSA Z243.4.1-1998 (and -1992). The delta that precedes the benchmark *has been simplified* for illustration here; a larger delta is required, mainly for special characters, for full conformance to this Canadian standard, and is given here as an example only, limited to what is required for the benchmark. For complete information the Canadian standard CAN/CSA Z243.4.1 should be consulted. The example's specifications are to be performed using the Common Template Table of annex A, with the following delta:

1. Processing properties:
```
order_start <TABLE>;forward;backward;forward;forward,position
```
2. Number of levels unchanged to 4.
3. No symbol change.
4. No other insertion, deletion or redefinition than:

   - æ sorted as if it were separate letters "ae" at level 1. The letters "ae" are distinguished at level 2 from the character "æ" and are sorted before it. Upper case is distinguished from lower case at level 3.

   - ð sorted as if it were the letter "d" at level 1. The letter "ð" is distinguished at level 2 from the letter "d" and is sorted after it. Upper case is distinguished from lower case at level 3.

   - þ sorted as if it were separate letters "th" at level 1. The letters "th" are distinguished at level 2 from the letter "þ" and are sorted before it. Upper case is distinguished from lower case at level 3.

Alternate formal ISO/IEC DTR 14652 tailoring equivalent

************* Ken to provide changes to this paragraph in pink ****************

```
% copy ISO14651_1999_TABLE1
reorder_after <SFFFF>
order_start forward;backward;forward;forward,position
reorder-after <U00C6>
<U00E6>  <S6CD><S72D>;<COMPAT><COMPAT>;<MIN><MIN>;IGNORE   % <ae>
<U00C6>  <S6CD><S72D>;<COMPAT><COMPAT>;<CAP><CAP>;IGNORE   % <AE>
reorder-after <U1E0E>
<U00F0>  <S705>;<COMPAT>;<MIN>;IGNORE                      % <d->
<U00D0>  <S705>;<COMPAT>;<CAP>";IGNORE                     % <D->
reorder-after <U2122>
<U00FE>  <S88B><S781>;<COMPAT><COMPAT>;<MIN><MIN>;IGNORE   % <th>
<U00DE>  <S88B><S781>;<COMPAT><COMPAT>;<CAP><CAP>;IGNORE   % <TH>
reorder-end
```

1 Unordered list (required test as per Canadian standard CAN/CSA Z243.4.1-1998 plus additions)

ou
lésé
péché
vice-président
9999
OÙ
haïe
coop
caennais
lèse
dû
air@@@
côlon
bohème
gêné
meðal
lamé
pêche
LÈS
vice versa
C.A.F.
Þorsmörk
cæsium
resumé
Bohémien
co-op
pêcher
les
CÔTÉ
résumé
Ålborg
cañon
du
haie
pécher
Mc Arthur
cote
colon
l'âme
resume
élève
Þorvarður
Canon
lame
Bohême
0000
relève
gène
casanier
élevé
COTÉ
relevé
Grossist
vice-presidents' offices
Copenhagen
côte
McArthur
Mc Mahon
Aalborg
Größe
vice-president's offices
cølibat
PÉCHÉ
COOP
@@@air
VICE-VERSA
gêne
CO-OP
révélé
révèle
çà et là
MacArthur
Noël
île
aïeul
Île d'Orléans
nôtre
notre
août
NOËL
@@@@@
L'Haÿ-les-Roses
CÔTE
COTE
côté
coté
aide
air
vice-president
modelé
Thorvardur
MODÈLE
maçon
MÂCON
pèche
pêché
medalovoïde
pechère
ode
péchère
œil

2 List with required results as per Canadian standard CAN/CSA Z243.4.1-1998

@@@@@
0000
9999
Aalborg
aide
aïeul
air
@@@air
air@@@
Ålborg
août
bohème
Bohême
Bohémien
caennais
cæsium
çà et là
C.A.F.
Canon
cañon
casanier
cølibat
colon
côlon
coop
co-op
COOP
CO-OP
Copenhagen
cote
COTE
côte
CÔTE
coté
COTÉ
côté
CÔTÉ
du
dû
élève
élevé
gène
gêne
gêné
Größe
Grossist
haie
haïe
île
Île d'Orléans
lame
l'âme
lamé
les
LÈS
lèse
lésé
L'Haÿ-les-Roses
MacArthur
MÂCON
maçon
medal
meðal
McArthur
Mc Arthur
Mc Mahon
MODÈLE
modelé
Noël
NOËL
notre
nôtre
ode
œil
ou
OÙ
ovoïde
pèche
pêche
péché
PÉCHÉ
pêché
pécher
pêcher
pechère
péchère
relève
relevé
resume
resumé
résumé
révèle
révélé
Þorsmörk
Thorvardur
Þorvarður
vice-president
vice-président
vice-president's offices
vice-presidents' offices
vice versa
VICE-VERSA

B.2 Example 2 – Danish delta and benchmark

The following is a Danish delta tailoring example. This formal specification corresponds to Danish standard DS 377 and to "Retskrivningsordbogen", the Danish orthography specification. The repertoire used assumes the exclusion of combining characters.

************ Ken to provide changes in the following pink lines **********

```
% This tailoring is in accordance with Danish Standard DS 377 (1980)
% and the Danish Orthography Dictionary (Retskrivningsordbogen, 1986).
% It is also in accordance with Greenlandic orthography.

% Define a collating element for a-ring.

collating-element <AA>

% Define collating elements for sequences of a+a

collating-element <A-A> from "<U0041><U0041>"
collating-element <A-a> from "<U0041><U0061>"
collating-element <a-A> from "<U0061><U0041>"
collating-element <a-a> from "<U0061><U0061>"

% Make capital letters sort before lowercase
% Note that this simple tailoring only impacts basic letters.
% To also make capital letters in compatibility characters sort
% before lowercase, a slightly more complex tailoring is required.

reorder-after <CAP>
<MIN>

% Reorder a-diaeresis, o-stroke, and a-ring at the end of the alphabet, after z.

reorder-after <S8E5> % ezh with curl (after z)
<S6D3> % ae
<S80D> % o-stroke
<AA>   % a-ring

% A list of reweighting statements to deal with specific
% Danish behavior. All of these define or redefine weight_list's,
% and so the entire block can simply be reordered after the
% last entry in the table.

reorder-after <SFFFF>

order_start forward;backward;forward;forward,position

% Space, hyphen-minus, and solidus are a primary weight
% before any letter or digit, with hyphen-minus and solidus
% given a secondary difference from the weight for space.

<U0020> <S209>;<BASE>;<MIN>;<U0020> % SPACE
<U002D> <S209>;"<BASE><VRNT1>";"<MIN><MIN>";<U002D> % HYPHEN-MINUS
<U002F> <S209>;"<BASE><VRNT2>";"<MIN><MIN>";<U002F> % SOLIDUS

% The letter kra (for Greenlandic) is equated to a lowercase q,
% with a secondary difference to distinguish it from q itself.

<U0138> <S829>;"<BASE><VRNT1>";"<MIN><MIN>";<U0138> % LATIN SMALL LETTER KRA

% The letter thorn is treated as a sequence of t + h, with a variant weight
% at the secondary level (comparable to the treatment of sharp-s).

<U00DE> "<S875><S773>";"<BASE><VRNT1><BASE>";"<CAP><MIN><CAP>";<U00DE> % LATIN CAPITAL LETTER
THORN
```

```
<U00FE> "<S875><S773>";"<BASE><VRNT1><BASE>";"<MIN><MIN><MIN>";<U00FE> % LATIN SMALL LETTER THORN

% The letters u-diaeresis and u-double-acute are given the same primary
% weight as y, with unique variant weights at the secondary level.

<U00DC> <S8BD>;"<BASE><VRNT1>";"<CAP><MIN>";<U00DC> % LATIN CAPITAL LETTER U WITH DIAERESIS
<U00FC> <S8BD>;"<BASE><VRNT1>";"<MIN><MIN>";<U00FC> % LATIN SMALL LETTER U WITH DIAERESIS
<U0170> <S8BD>;"<BASE><VRNT2>";"<CAP><MIN>";<U0170> % LATIN CAPITAL LETTER U WITH DOUBLE ACUTE
<U0171> <S8BD>;"<BASE><VRNT2>";"<MIN><MIN>";<U0171> % LATIN SMALL LETTER U WITH DOUBLE ACUTE

% The letters o-diaeresis and o-double-acute are given the same primary
% weight as o-slash, with unique variant weights at the secondary level.

<U00D6> <S80D>;"<BASE><VRNT1>";"<CAP><MIN>";<U00D6> % LATIN CAPITAL LETTER O WITH DIAERESIS
<U00F6> <S80D>;"<BASE><VRNT1>";"<MIN><MIN>";<U00F6> % LATIN SMALL LETTER O WITH DIAERESIS
<U0150> <S80D>;"<BASE><VRNT2>";"<CAP><MIN>";<U0150> % LATIN CAPITAL LETTER O WITH DOUBLE ACUTE
<U0151> <S80D>;"<BASE><VRNT2>";"<MIN><MIN>";<U0151> % LATIN SMALL LETTER O WITH DOUBLE ACUTE

% The letter a-ring is weighted following the letter o-slash (see above)

<U00C5> <AA>;<BASE>;<CAP>;<U00C5> % LATIN CAPITAL LETTER A WITH RING
<U00E5> <AA>;<BASE>;<CAP>;<U00E5> % LATIN SMALL LETTER A WITH RING
<U01FA> <AA>;"<BASE><AIGUT>";"<CAP><MIN>";<U01FA> % LATIN CAPITAL LETTER A WITH RING ABOVE AND
ACUTE
<U01FB> <AA>;"<BASE><AIGUT>";"<MIN><MIN>";<U01FB> % LATIN SMALL LETTER A WITH RING ABOVE AND
ACUTE

% The sequences of letters a+a are weighted a secondary variants of a-ring

<A-A> <AA>;"<BASE><VRNT1>";"<CAP><CAP>";<U0041><U0041> % AA
<A-a> <AA>;"<BASE><VRNT1>";"<CAP><MIN>";<U0041><U0061> % Aa
<a-A> <AA>;"<BASE><VRNT1>";"<MIN><CAP>";<U0061><U0041> % aA
<a-a> <AA>;"<BASE><VRNT1>";"<MIN><MIN>";<U0061><U0061> % aa

reorder-end
```

% End of the Danish tailoring

Benchmark 2 for Danish

A/S
ANDRE
ANDRÉ
ANDREAS
AS
CA
ÇA
CB
ÇC
DA
ĐA
DB
ĐC
DSB
D.S.B.
DSC
EKSTRA-ARBEJDE
EKSTRABUD
EKSTRAARBEJDE
HØST
HAAG
HÅNDBOG
HAANDVÆRKSBANKEN
Karl
karl
NIELS JØRGEN
NIELS-JØRGEN
NIELSEN
RÉE, A
REE, B
RÉE, L
REE, V
SCHYTT, B
SCHYTT, H
SCHÜTT, H
SCHYTT, L
SCHÜTT, M
ß
SS
SSA
STORE VILDMOSE
STOREKÆR
STORM PETERSEN
STORMLY
THORVALD
THORVARDUR
ÞORVARÐUR
THYGESEN
VESTERGÅRD, A
VESTERGAARD, A
VESTERGÅRD, B
ÆBLE
ÄBLE
ØBERG
ÖBERG

B.3 Example 3 – Reversing the order of lower case and upper case letters

The following is a simple tailoring example to show how to reverse
the order of uppercase versus lowercase from the order specified
in the Common Tailorable Template Table.

********* Ken to provide changes in the following pink lines ***********

```
% Make uppercase letters sort before lowercase
% and scanning of accents done forward at level 2.
% To do this correctly,
% first an order_start is inserted to make the delta conformant.
% Then, the entire range of tertiary weight symbols
% <MIN>..<CIRCLE> are moved after <CIRCLECAP>, so that they order after
% <CAP> <WIDECAP> <COMPATCAP <FONTCAP> <CIRCLECAP> in the same
% relative order with respect to themselves. This has the effect of
% also making all the compatibility uppercase letters sort before
% their respective compatibility lowercase letters. (For example,
% U+24B6 CIRCLED LATIN CAPITAL LETTER A will sort before
% U+24D0 CIRCLED LATIN SMALL LETTER A.

reorder_after <SFFFF>
order_start forward;forward;forward;forward,position
reorder-after <CIRCLECAP>
<MIN>
<WIDE>
<COMPAT>
<FONT>
<CIRCLE>

reorder-end

% End of the uppercase/lowercase tailoring
```

B.4 Cyrillic (issue)

*************** Ken, do we keep this finally, after discussion *****************

IMPORTANT NOTE.

> *In accordance with the disposition of comments on the second FCD of ISO/IEC 14651, there is a known issue on Cyrillic in the Common Template Table that needs to be solved before the DIS. The common template table is mostly self-generated from character properties described in tables detained by the Unicode consortium. This led to known and manageable mismatches between this table and actual Cyrillic sorting practice in particular in Russia.*
>
> *On one hand it is known that the current template table for Cyrillic does not sort Russian correctly for some letters affected by a basic diacritic. The following delta makes the appropriate corrections.*
>
> *On the other hand defining equivalence of sorting for combining sequences is still an issue whose solution is known but which was not solved on time to obey to optimum FCD ballot deadline constraints.*
>
> *It is the intent that after the third FCD ballot, this issue will be solved by consensus before the DIS is published. This is expected to be the main issue remaining before the DIS ballot and the ballot period should allow sufficient time for a discussion leading to the optimum solution, whether this will be done via a delta of by manual modification of the self-generated table*

```
reorder-after <U04C0>

<U04D1> <S95B>;<BASE>;<MIN>;<U04D1> % CYRILLIC SMALL LETTER A WITH BREVE
"<U0430><U0306>" <S95B>;<BASE>;<MIN>;"<U0430><U0306>" % CYRILLIC SMALL LETTER A with COMBINING BREVE
<U04D0> <S95B>;<BASE>;<CAP>;<U04D0> % CYRILLIC CAPITAL LETTER A WITH BREVE
"<U0410><U0306>" <S95B>;<BASE>;<CAP>;"<U0410><U0306>" % CYRILLIC CAPITAL LETTER A with COMBINING BREVE
<U04D3> <S95C>;<BASE>;<MIN>;<U04D3> % CYRILLIC SMALL LETTER A WITH DIAERESIS
"<U0430><U0308>" <S95C>;<BASE>;<MIN>;"<U0430><U0308>" % CYRILLIC SMALL LETTER A with COMBINING DIAERESIS
<U04D2> <S95C>;<BASE>;<CAP>;<U04D2> % CYRILLIC CAPITAL LETTER A WITH DIAERESIS
"<U0410><U0308>" <S95C>;<BASE>;<CAP>;"<U0410><U0308>" % CYRILLIC CAPITAL LETTER A with COMBINING DIAERESIS

<U04DB> <S95F>;<BASE>;<MIN>;<U04DB> % CYRILLIC SMALL LETTER SCHWA WITH DIAERESIS
"<U04D9><U0308>" <S95F>;<BASE>;<MIN>;"<U04D9><U0308>" % CYRILLIC SMALL LETTER SCHWA with COMBINING DIAERESIS
<U04DA> <S95F>;<BASE>;<CAP>;<U04DA> % CYRILLIC CAPITAL LETTER SCHWA WITH DIAERESIS
"<U04D8><U0308>" <S95F>;<BASE>;<CAP>;"<U04D8><U0308>" % CYRILLIC CAPITAL LETTER SCHWA with COMBINING
DIAERESIS

<U0453> <S983>;<BASE>;<MIN>;<U0453> % CYRILLIC SMALL LETTER GJE
"<U0433><U0301>" <S983>;<BASE>;<MIN>;"<U0433><U0301>" % CYRILLIC SMALL LETTER GHE with COMBINING ACUTE ACCENT
<U0403> <S983>;<BASE>;<CAP>;<U0403> % CYRILLIC CAPITAL LETTER GJE
"<U0413><U0301>" <S983>;<BASE>;<CAP>;"<U0413><U0301>" % CYRILLIC CAPITAL LETTER GHE with COMBINING ACUTE
ACCENT

<U04D7> <S98B>;<BASE>;<MIN>;<U04D7> % CYRILLIC SMALL LETTER IE WITH BREVE
"<U0435><U0306>" <S98B>;<BASE>;<MIN>;"<U0435><U0306>" % CYRILLIC SMALL LETTER IE with COMBINING BREVE
<U04D6> <S98B>;<BASE>;<CAP>;<U04D6> % CYRILLIC CAPITAL LETTER IE WITH BREVE
"<U0415><U0306>" <S98B>;<BASE>;<CAP>;"<U0415><U0306>" % CYRILLIC CAPITAL LETTER IE with COMBINING BREVE

<U04C2> <S993>;<BASE>;<MIN>;<U04C2> % CYRILLIC SMALL LETTER ZHE WITH BREVE
"<U0436><U0306>" <S993>;<BASE>;<MIN>;"<U0436><U0306>" % CYRILLIC SMALL LETTER ZHE with COMBINING BREVE
<U04C1> <S993>;<BASE>;<CAP>;<U04C1> % CYRILLIC CAPITAL LETTER ZHE WITH BREVE
"<U0416><U0306>" <S993>;<BASE>;<CAP>;"<U0416><U0306>" % CYRILLIC CAPITAL LETTER ZHE with COMBINING BREVE
<U04DD> <S994>;<BASE>;<MIN>;<U04DD> % CYRILLIC SMALL LETTER ZHE WITH DIAERESIS
"<U0436><U0308>" <S994>;<BASE>;<MIN>;"<U0436><U0308>" % CYRILLIC SMALL LETTER ZHE with COMBINING DIAERESIS
<U04DC> <S994>;<BASE>;<CAP>;<U04DC> % CYRILLIC CAPITAL LETTER ZHE WITH DIAERESIS
"<U0416><U0308>" <S994>;<BASE>;<CAP>;"<U0416><U0308>" % CYRILLIC CAPITAL LETTER ZHE with COMBINING DIAERESIS

<U04DF> <S99B>;<BASE>;<MIN>;<U04DF> % CYRILLIC SMALL LETTER ZE WITH DIAERESIS
"<U0437><U0308>" <S99B>;<BASE>;<MIN>;"<U0437><U0308>" % CYRILLIC SMALL LETTER ZE with COMBINING DIAERESIS
<U04DE> <S99B>;<BASE>;<CAP>;<U04DE> % CYRILLIC CAPITAL LETTER ZE WITH DIAERESIS
"<U0417><U0308>" <S99B>;<BASE>;<CAP>;"<U0417><U0308>" % CYRILLIC CAPITAL LETTER ZE with COMBINING DIAERESIS

<U04E5> <S9A7>;<BASE>;<MIN>;<U04E5> % CYRILLIC SMALL LETTER I WITH DIAERESIS
"<U0438><U0308>" <S9A7>;<BASE>;<MIN>;"<U0438><U0308>" % CYRILLIC SMALL LETTER I with COMBINING DIAERESIS
<U04E4> <S9A7>;<BASE>;<CAP>;<U04E4> % CYRILLIC CAPITAL LETTER I WITH DIAERESIS
"<U0418><U0308>" <S9A7>;<BASE>;<CAP>;"<U0418><U0308>" % CYRILLIC CAPITAL LETTER I with COMBINING DIAERESIS
<U04E3> <S9A8>;<BASE>;<MIN>;<U04E3> % CYRILLIC SMALL LETTER I WITH MACRON
```

```
"<U0438><U0304>" <S9A8>;<BASE>;<MIN>;"<U0438><U0304>" % CYRILLIC SMALL LETTER I with COMBINING MACRON
<U04E2> <S9A8>;<BASE>;<CAP>;<U04E2> % CYRILLIC CAPITAL LETTER I WITH MACRON
"<U0418><U0304>" <S9A8>;<BASE>;<CAP>;"<U0418><U0304>" % CYRILLIC CAPITAL LETTER I with COMBINING MACRON
<U0457> <S9AB>;<BASE>;<MIN>;<U0457> % CYRILLIC SMALL LETTER YI
"<U0456><U0308>" <S9AB>;<BASE>;<MIN>;"<U0456><U0308>" % CYRILLIC SMALL LETTER BYELORUSSIAN-UKRAINIAN I with COMBINING
DIAERESIS
<U0407> <S9AB>;<BASE>;<CAP>;<U0407> % CYRILLIC CAPITAL LETTER YI
"<U0406><U0308>" <S9AB>;<BASE>;<CAP>;"<U0406><U0308>" % CYRILLIC CAPITAL LETTER BYELORUSSIAN-UKRAINIAN I with
COMBINING DIAERESIS
<U0439> <S9AC>;<BASE>;<MIN>;<U0439> % CYRILLIC SMALL LETTER SHORT I
"<U0438><U0306>" <S9AC>;<BASE>;<MIN>;"<U0438><U0306>" % CYRILLIC SMALL LETTER I with COMBINING BREVE
<U0419> <S9AC>;<BASE>;<CAP>;<U0419> % CYRILLIC CAPITAL LETTER SHORT I
"<U0418><U0306>" <S9AC>;<BASE>;<CAP>;"<U0418><U0306>" % CYRILLIC CAPITAL LETTER I with COMBINING BREVE

<U04E7> <S9EC>;<BASE>;<MIN>;<U04E7> % CYRILLIC SMALL LETTER O WITH DIAERESIS
"<U043E><U0308>" <S9EC>;<BASE>;<MIN>;"<U043E><U0308>" % CYRILLIC SMALL LETTER O with COMBINING DIAERESIS
<U04E6> <S9EC>;<BASE>;<CAP>;<U04E6> % CYRILLIC CAPITAL LETTER O WITH DIAERESIS
"<U041E><U0308>" <S9EC>;<BASE>;<CAP>;"<U041E><U0308>" % CYRILLIC CAPITAL LETTER O with COMBINING DIAERESIS
<U04EB> <S9F0>;<BASE>;<MIN>;<U04EB> % CYRILLIC SMALL LETTER BARRED O WITH DIAERESIS
"<U04E9><U0308>" <S9F0>;<BASE>;<MIN>;"<U04E9><U0308>" % CYRILLIC SMALL LETTER BARRED O with COMBINING
DIAERESIS
<U04EA> <S9F0>;<BASE>;<CAP>;<U04EA> % CYRILLIC CAPITAL LETTER BARRED O WITH DIAERESIS
"<U04E8><U0308>" <S9F0>;<BASE>;<CAP>;"<U04E8><U0308>" % CYRILLIC CAPITAL LETTER BARRED O with COMBINING
DIAERESIS

<U045C> <SA13>;<BASE>;<MIN>;<U045C> % CYRILLIC SMALL LETTER KJE
"<U043A><U0301>" <SA13>;<BASE>;<MIN>;"<U043A><U0301>" % CYRILLIC SMALL LETTER KA with COMBINING ACUTE ACCENT
<U040C> <SA13>;<BASE>;<CAP>;<U040C> % CYRILLIC CAPITAL LETTER KJE
"<U041A><U0301>" <SA13>;<BASE>;<CAP>;"<U041A><U0301>" % CYRILLIC CAPITAL LETTER KA with COMBINING ACUTE
ACCENT

<U045E> <SA17>;<BASE>;<MIN>;<U045E> % CYRILLIC SMALL LETTER SHORT U
"<U0443><U0306>" <SA17>;<BASE>;<MIN>;"<U0443><U0306>" % CYRILLIC SMALL LETTER U with COMBINING BREVE
<U040E> <SA17>;<BASE>;<CAP>;<U040E> % CYRILLIC CAPITAL LETTER SHORT U
"<U0423><U0306>" <SA17>;<BASE>;<CAP>;"<U0423><U0306>" % CYRILLIC CAPITAL LETTER U with COMBINING BREVE
<U04F1> <SA18>;<BASE>;<MIN>;<U04F1> % CYRILLIC SMALL LETTER U WITH DIAERESIS
"<U0443><U0308>" <SA18>;<BASE>;<MIN>;"<U0443><U0308>" % CYRILLIC SMALL LETTER U with COMBINING DIAERESIS
<U04F0> <SA18>;<BASE>;<CAP>;<U04F0> % CYRILLIC CAPITAL LETTER U WITH DIAERESIS
"<U0423><U0308>" <SA18>;<BASE>;<CAP>;"<U0423><U0308>" % CYRILLIC CAPITAL LETTER U with COMBINING DIAERESIS
<U04F3> <SA19>;<BASE>;<MIN>;<U04F3> % CYRILLIC SMALL LETTER U WITH DOUBLE ACUTE
"<U0443><U030B>" <SA19>;<BASE>;<MIN>;"<U0443><U030B>" % CYRILLIC SMALL LETTER U with COMBINING DOUBLE ACUTE
ACCENT
<U04F2> <SA19>;<BASE>;<CAP>;<U04F2> % CYRILLIC CAPITAL LETTER U WITH DOUBLE ACUTE
"<U0423><U030B>" <SA19>;<BASE>;<CAP>;"<U0423><U030B>" % CYRILLIC CAPITAL LETTER U with COMBINING DOUBLE ACUTE
ACCENT
<U04EF> <SA1A>;<BASE>;<MIN>;<U04EF> % CYRILLIC SMALL LETTER U WITH MACRON
"<U0443><U0304>" <SA1A>;<BASE>;<MIN>;"<U0443><U0304>" % CYRILLIC SMALL LETTER U with COMBINING MACRON
<U04EE> <SA1A>;<BASE>;<CAP>;<U04EE> % CYRILLIC CAPITAL LETTER U WITH MACRON
"<U0423><U0304>" <SA1A>;<BASE>;<CAP>;"<U0423><U0304>" % CYRILLIC CAPITAL LETTER U with COMBINING MACRON
<U04AF> <SA1C>;<BASE>;<MIN>;<U04AF> % CYRILLIC SMALL LETTER STRAIGHT U
<U04AE> <SA1C>;<BASE>;<CAP>;<U04AE> % CYRILLIC CAPITAL LETTER STRAIGHT U

<U04F5> <SA4F>;<BASE>;<MIN>;<U04F5> % CYRILLIC SMALL LETTER CHE WITH DIAERESIS
"<U0447><U0308>" <SA4F>;<BASE>;<MIN>;"<U0447><U0308>" % CYRILLIC SMALL LETTER CHE with COMBINING DIAERESIS
<U04F4> <SA4F>;<BASE>;<CAP>;<U04F4> % CYRILLIC CAPITAL LETTER CHE WITH DIAERESIS
"<U0427><U0308>" <SA4F>;<BASE>;<CAP>;"<U0427><U0308>" % CYRILLIC CAPITAL LETTER CHE with COMBINING DIAERESIS

<U04F9> <SA78>;<BASE>;<MIN>;<U04F9> % CYRILLIC SMALL LETTER YERU WITH DIAERESIS
"<U044B><U0308>" <SA78>;<BASE>;<MIN>;"<U044B><U0308>" CYRILLIC SMALL LETTER YERU with COMBINING DIAERESIS
<U04F8> <SA78>;<BASE>;<CAP>;<U04F8> % CYRILLIC CAPITAL LETTER YERU WITH DIAERESIS
"<U042B><U0308>" <SA78>;<BASE>;<CAP>;"<U042B><U0308>" % CYRILLIC CAPITAL LETTER YERU with COMBINING DIAERESIS

<U0477> <SAAF>;<BASE>;<MIN>;<U0477> % CYRILLIC SMALL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT
"<U0475><U030F>" <SAAF>;<BASE>;<MIN>;"<U0475><U030F>" % CYRILLIC SMALL LETTER IZHITSA with COMBINING DOUBLE
GRAVE ACCENT
<U0476> <SAAF>;<BASE>;<CAP>;<U0476> % CYRILLIC CAPITAL LETTER IZHITSA WITH DOUBLE GRAVE ACCENT
"<U0474><U030F>" <SAAF>;<BASE>;<CAP>;"<U0474><U030F>" % CYRILLIC CAPITAL LETTER IZHITSA with COMBINING DOUBLE
GRAVE ACCENT

end-reorder
```

C -- Preparation (informative)

C.1 General considerations

Preparation is necessary only for modification and/or duplication of original strings to render them context-independent prior to the comparison phase. Examples are:

- duplication of a string such as "41" for as it is spelled out in different languages (Irish Gaelic, German, English, and French):

  daichead a haon

  einundvierzig

  forty-one

  quarante-et-un

- removal or rotation of characters that are a nuisance for special requirements of ordering; for example, removing articles in sorting book names as in

  Tale of two cities, A

- transformation of abbreviated data into a fuller form; for example, transformation of "McArthur" to give "MacArthur"

- transformation of numbers so that the result will be ordered in numerical order, as opposed to positional order (see below). Numeric ordering is particularly delicate and requires special consideration in many cases.

C.2 Handling of numeral substrings in collation

A numeral is a string representing a number. The examples here deal with numerals which represent values in *R*, the real numbers, or subsets of *R*, as these have a predetermined order. Only decimal numerals are dealt with in the examples given here.

The presentation below will first give positional system decimal numerals for natural numbers using the digits 0-9. It will progress to numerals for whole numbers, numerals with a fraction part, a fraction part and an exponent. There is also a brief discussion on numerals with digits from other scripts, scripts which sometimes uses another syntax with digits for numerals (such as Hàn numerals), and Roman numerals. There are circumstances where digits do not represent numeral values, such as in part numbers. These cases are not discussed below. Caution: preparatory steps have undesirable consequences in some cases such as the ordering of telephone numbers in telephone books, and should be avoided in those cases.

C.2.1 Handling of 'ordinary' numerals for natural numbers

The Common Template Table has no means of sorting strings with numbers in such a way that the resulting order reflects the number values represented by the numerals. For example, given the following randomly-arranged strings:

Release 1
Release 20

Release 12
Release 2
Release 9

the method described in the this International Standard yields the following list of “sorted” strings:

Release 1
Release 12
Release 2
Release 20
Release 9

(It is sufficient simply to look positionally at just the first digit in each numeral to see why this ordering results.) A more acceptable ordering is:

Release 1
Release 2
Release 9
Release 12
Release 20

The Common Template Table defined in this International Standard cannot be tailored to give this result. However, preparation can be done prior to the basic collation step to achieve the desired results when numeric value order is desired. The prepared strings are normally not presented to the user; only the original strings are. The prepared strings are normally only used for the collation key construction. A simple, but not very general, way of preparing numerals for natural ordering is to pad them with zeroes to a given number of digits. If one pads the numerals in our original example strings up to three digits, the following will result:

Release 001
Release 020
Release 012
Release 002
Release 009

Using the Common Template Table defined in this International Standard one then obtains the strings in a better order (here showing the strings as they are after preparation, which are normally not shown in the result):

Release 001
Release 002
Release 009
Release 012
Release 020

However, there are two problems with this approach:

1. One must determine beforehand a (usually small) number of digits to pad up to. If the number of digits to pad up to is too large, the strings after preparation can become rather long, especially if there are several numerals in each string. If the number of digits to pad up to is too small, however, the risk is greater that there are actually occurring numerals with more digits than one has padded up to, which results in partially getting back to the original situation, where the numerals’ values are not taken entirely into account.

2. Determinacy is lost, if some of the original numerals were already partially zero-padded. For example, if the original strings were:

```
Release 01
Release 1
```

the strings after preparation are identical, and the end result (as the user would normally see it) could be either

```
Release 01
Release 1
```

or

```
Release 1
Release 01
```

and the relative order may come out differently for different occurrences of numerals, or different runs of the collation process applying the same rules. Indeterminacy in collation is not desirable.

There are many ways to deal with these problems. The following is one such way.

To each maximal digit subsequence prepend a fixed-number-of-digits numeral which represents the original number of digits in the numeral. For most cases a two-digit count would suffice (allowing up to 99 digits in the original integer numerals). For example, given the original strings:

```
Release 1
Release 01
Release 20
Release 12
Release 2
Release 09
Release 9
```

One obtains after this preparation the following strings:

```
Release 011
Release 0201
Release 0220
Release 0212
Release 012
Release 0209
Release 019
```

Which would be collated by the basic mechanism of this International Standard to:

```
Release 011
Release 012
Release 019
Release 0201
Release 0209
Release 0212
Release 0220
```

As normally presented to the user:

```
Release 1
Release 2
Release 9
Release 01
Release 09
Release 12
Release 20
```

This particular method puts numerals with a like original number of digits close to each other, even if the actual value represented is smaller due to the original zero-padding. If the represented *values* should be kept close together, one should instead duplicate the numeral: first a count of digits for the leading-zero-stripped numeral, the leading-zero-stripped numeral itself, followed by the original numeral. The duplication is needed to get determinacy relative to the original strings. For example, using the same original strings as above:

```
Release 011 1
Release 011 01
Release 0220 20
Release 0212 12
Release 012 2
Release 019 09
Release 019 9
```

Which would be collated by the basic mechanism of this standard to:

```
Release 011 01
Release 011 1
Release 012 2
Release 019 09
Release 019 9
Release 0212 12
Release 0220 20
```

As normally presented to the user:

```
Release 01
Release 1
Release 2
Release 09
Release 9
Release 12
Release 20
```

The originally zero-padded numerals consistently come before the numeral without (or with less) original zero-padding. The preparation processing could move the original numerals (in order of occurrence) to the very end of each string, if one wants to give the original zero-padding lesser significance than the text following the numerals.

The presence of several natural numerals in each string causes no additional problem.

Taking care of the natural number numerals is in most cases sufficient, and it is recommended that it be included as part of the usual preparation of strings to be collated. However such preparation is not required by this International Standard.

C.2.2 Handling of positional numerals in other scripts

ISO/IEC 10646 encodes decimal digits for a number of scripts. In most cases these are used in a positional system, just like 0-9 usually are. However, one should not regard a sequence of numerals mixed from different scripts as a single numeral; rather, one should consider each maximal substring of digits of the same script to be a numeral.

C.2.3 Handling of other non-pure positional system numerals or non-positional system numerals (e.g. Roman numerals)

Chinese and some other languages can use decimal digits (in the Hàn script, for instance) interspersed with ideographs for “one thousand”, “ten”, etc. If such numerals are to be collated according to the value they represent, one can proceed as above, adding a step just after the initial duplication: convert the copy to the corresponding positional system numeral in the syntax used here for whole numerals.

Roman numerals, if handled, can be handled in a similar fashion to that described above. Duplicate, and replace the first copy with the same natural number expressed in the decimal positional system. E.g. “Louis V”, where the V is determined to be a Roman numeral, can be modified to “Louis 5 V”.

Caveat: In this case human interactive intervention or an expert system may be required, as in the following example involving the French language: CHAPITRE DIX might mean CHAPTER 10 or CHAPTER 509 ("dix" is the French word for 10, it is also the Roman numeral for 509).

C.2.4 Handling of numerals for whole numbers

If negative whole numbers are also to be sorted according to their value, there are a number of issues to be considered. Most frequently, negative whole values are given numerals that begin with a negation sign. The negation sign may be HYPHEN-MINUS U+002D (caveat: this character may represent a true hyphen, rather than a negation), or MINUS SIGN U+2212. But there are other conventions also, like using a SOLIDUS U+002F or a PERCENT SIGN U+0025 to indicate negativeness; or the negation indicator can come *after* the digits rather than before; or negativeness can be indicated by putting the digits between parenthesis, and/or putting the digits in a contrasting color (often red). In the examples here, only the case that negativeness is indicated by an immediately prepended MINUS SIGN is dealt with. Positiveness is indicated by either the absence of a MINUS SIGN, or the presence of a PLUS SIGN U+002B.

```
Temperature: –9 °C
Temperature: 0 °C
Temperature: –14 °C
Temperature: 05 °C
Temperature: +5 °C
Temperature: –0 °C
Temperature: –09 °C
Temperature: 105 °C
Temperature: +05 °C
Temperature: 5 °C
```

One preparation to get an acceptable and determinate order for numerals (in this syntax) for whole numbers is as follows (actual implementations should do something equivalent, but more efficient):

3. Duplicate the numerals in the string (including sign indications), putting the ‘original’ ones (not to be touched by the following steps) in order of original occurrence at the end or the string, leaving the copies at the original positions. This step ensures determinacy.

4. Ensure that all of the copies have an explicit initial sign indicator.

5. Remove leading zeroes in the copies of the numerals (systematically either leaving one zero digit for zero or representing 0 by the empty string of digits); alternatively, let all numeral copies have exactly one leading zero.

6. Between the sign indicator and the digits in the copies of the numerals, insert a (two-digit) count of how many digits there were (after removing the leading zeroes).

7. Do 9's complement on each digit in each copy of a negated numeral. 9's complement of a digit that individually represents the value *x*, is 9–*x*. That is, 9's complement of 0 is 9, of 9 is 0, of 5 is 4, etc.

8. Done with this (part of the) preparation.

For the basic collation step, use a tailoring of the template given in this standard, namely, a tailoring where the PLUS SIGN and the MINUS SIGN are significant at the same level as the digits, and where the MINUS SIGN has less weight than the PLUS SIGN. (In the example below, it is assumed that the weight of PLUS SIGN is less than the weight of 0, but this is not a prerequisite for getting an acceptable ordering.)

Our example strings after this prehanding:

```
Temperature: –980 °C –9
Temperature: +00 °C 0
Temperature: –9785 °C –14
Temperature: +015 °C 05
Temperature: +015 °C +5
Temperature: –99 °C –0
Temperature: –980 °C –09
Temperature: +03105 °C 105
Temperature: +015 °C +05
Temperature: +015 °C 5
```

Sort these, using the basic mechanism of this standard:

```
Temperature: –9785 °C –14
Temperature: –980 °C –09
Temperature: –980 °C –9
Temperature: –99 °C –0
Temperature: +00 °C 0
Temperature: +015 °C +05
Temperature: +015 °C +5
Temperature: +015 °C 05
Temperature: +015 °C 5
Temperature: +03105 °C 105
```

As presented to the user:

```
Temperature: –14 °C
Temperature: –09 °C
Temperature: –9 °C
Temperature: –0 °C
Temperature: 0 °C
Temperature: +05 °C
Temperature: +5 °C
Temperature: 05 °C
```

Temperature: 5 °C
Temperature: 105 °C

This preparation results in a determinate ordering of strings which may have numerals for whole numbers in them (also if there are several such numerals in some of the strings), such that the numerals are ordered according to the integer value they represent.

The process for other syntaxes for whole numbers can be similar. Just add a step to convert the copies to the syntax used here for whole numbers.

This technique for handling negative numerals can be used also for numerals with a fractional part, and so on (see below).

C.2.5 Handling of positive positional numerals with fractional parts

The method presented above can easily be adapted to the case where fraction parts may occur and are to be taken into account. A problem is, however, that the characters often used to delimit the integer part from the fraction part are also used for other purposes. The separator character is generally either FULL STOP U+002E, or COMMA U+002C. These characters also have other uses, also in conjunction with digits.

For the example, assume that FULL STOP is used (only) as a fraction part delimiter.

Do as above, but count only the digits in the integer part of the numeral for the count of digits to be prepended. The fraction part delimiter character can be removed.

For example:

–12.34

12.34

3.1415

3.14

After preparation:

–978765 –12.34

+021234 12.34

+013.1415 3.1415

+01314 3.14

After sorting:

–978765 –12.34

+01314 3.14

+0131415 3.1415

+021234 12.34

As presented to the user:

–12.34

3.14

3.1415

12.34

C.2.6 Handling of positive positional numerals with fraction parts and exponent parts

For very large, or very small, values, one often uses formats like 2.5*10^{7} (to illustrate just one possible way of writing these for the purposes of the examples here). Here there is already an exponent, which must be combined with the “number of integer part digits” (here: digits before the decimal point), by adding those two numbers to get a resulting fixed-number-of-digits exponent to prepend just before the first digit. For this example, with a three-digit exponent: we get +00825. One problem here is that the resulting exponent may be negative. To handle this, use an exponent bias. For a three-digit exponent a bias of 500 may be suitable, which gives us for this example numeral: +50825, and for the numeral 2.5*10$^{–7}$ we get +49425. Negative values are handled as before, with 9’s complement. –2.5*10^{7} gives –49174, and –2.5*10$^{–7}$ gives –50574.

This method should be familiar to anyone with knowledge about (radix 10) floating point arithmetic.

Thus:

2.5*10$^{–7}$
–2.5*10^{7}
2.5*10^{7}
–2.5*10$^{–7}$

After preparation (including a duplicate of the original, for determinacy):

+49425 2.5*10$^{–7}$
–49174 ¯2.5*10^{7}
+50825 2.5*10^{7}
–50574 ¯2.5*10$^{–7}$

After sorting:

–49174 ¯2.5*10^{7}
–50574 ¯2.5*10$^{–7}$
+49425 2.5*10$^{–7}$
+50825 2.5*10^{7}

As presented to the user:

–2.5*10^{7}
–2.5*10$^{–7}$
2.5*10$^{–7}$
2.5*10^{7}

C.2.8 Handling of date and time of day indications

Going a bit beyond plain numerals, date and time-of-day indications often employ numerals (as well as names for months, weekdays, etc.) for the parts of the date and time-of-day indication. It is not uncommon to want to sort this kind of information also when it occurs within strings.

The preparation needed to obtain date and time-of-day indications, of some predetermined syntaxes, sorted according to point in time is similar to what has been described above.

1. Duplicate all date and time-of-day indications to maintain determinacy of collation when the original strings differ, but point in time identical. Leave the originals at the end of the strings, untouched by the following steps.
2. Convert the copies of the date and time indications to the same calendar system, if there are several calendar (sub)systems used and handled. The calendar (sub)system converted to, must be suitable

for being able to get proper time order. We will here use the Gregorian calendar system and the subsystem of year, month, day-of-month.

3. Put the date and time-of-day elements in order of decreasing significance (to the resolution taken into account). Full year, month, day-of-month, hour, minute, second, fraction of second.
4. Use a 24-hour/day clock for the time-of-day indications. Remove A.M. or P.M. indications, if present and handled, in the date-time indication copies.
5. Use the UTC time zone for the date and time-of-day indications. Remove time zone indications, if present, in the date-time indication copies.
6. Use month numbers, rather than month names. Use two digits each for month, day-of-month, hour, minute, second.
7. Use full year number representation, as many digits as needed. Take abbreviations into account so that the full year number is used. E.g. '98' might denote year 98 or year 1998, or 1898, etc. No indeterminacy regarding year due to abbreviations like these may be present after the preparation step.
8. For years AD, use an initial PLUS SIGN. For years BC, use an initial MINUS SIGN. Remove the original AD or BC indication from the copies. (To nitpick, year *n* BC should be represented by year (1–*n*), which is less or equal to zero if *n* is positive.)
9. For the year indications, insert between the sign indication and the first digit for the year indication a digit telling how many digits there are in the full year indication. One digit for this should suffice.
10. For negative years, replace the each digit in the year indication (including the digit telling the number of digits in the original full year indication) with its 9's complement digit.
11. Make sure the textual format for all of the date indication copies is the same (paying attention to hyphens, spaces, etc.). (This is most easily accomplished by printing them in the same format from an internal, non-string, representation.)
12. Alternatively, use a number indicating the point of time on a linear time scale (for example, hours, milliseconds, or days from a predetermined point in time), to the resolution desired, and handle this as an ordinary numeral (see above).
13. Done with this (part of the) preparation.
14. For the basic collation step, use a tailoring of the template given in this standard. Use a tailoring where the PLUS SIGN and the MINUS SIGN are significant at the same level as the digits, and where the MINUS SIGN has less weight than the PLUS SIGN.

For example:

Dated: July 19, 1955, at 1 p.m. GMT
Dated: January, 20 BC
Dated: Sept. 20, 1995, at 1 p.m. PST
Dated: 11-june/345 AD

After preparation:

Dated: +41955-07-19T13:00Z July 19, 1955, at 1 p.m. GMT
Dated: –780-01 January, 20 BC
Dated: +1995-09-20T10:00Z Sept. 20, 1995, at 1 p.m. PST
Dated: +3345-06-11 11-june/345 AD

After sorting:

Dated: –780-01 January, 20 BC
Dated: +3345-06-11 11-june/345 AD
Dated: +41955-07-19T13:00Z July 19, 1955, at 1 p.m. GMT
Dated: +41995-09-20T10:00Z Sept. 20, 1995, at 1 p.m. PST

As presented to the user:

Dated: January, 20 BC
Dated: 11-june/345 AD
Dated: July 19, 1955, at 1 p.m. GMT
Dated: Sept. 20, 1995, at 1 p.m. PST

C.2.9 Making numbers less significant than letters

In many cases numerals preceding letters should be considered as less significant than the following alphabetic part. But the Common Template Table specifies digits to be level 1 significant. To make numerals less significant than letters, either tailor the weight table so that numerals are ignored at level 1 (but significant at level 2 or 3), or alternatively leave them significant at level 1, but prepare the strings so that numerals are moved to the end of the string or moved to a less significant field. When doing such a move, one must pay attention not to map different strings to identical strings (or identical string fields), so that determinacy is maintained (see C.2.10).

Some examples where it is appropriate to consider numerals as less significant than letters: Street or block names with one or more numbers to indicate where in the street/block, if that/those number(s) precede the street or block name (common for example in the US and in Japan); chemical compound names which have prepended numerals, e.g., 1,2-diclorobenzol.

C.2.10 Maintaining determinacy

As noted above in several cases, part of the string has been duplicated to maintain determinacy in collation, when the original strings are different, but when preparation may otherwise turn different strings into identical strings.

This method of duplication for determinacy can be used more generally, so if there are several preparations affecting different parts of the strings, one may simply duplicate the original strings to begin with, and only perform the preparation (without additional duplication) on the first half of the “doubled” string.

One disadvantage with just concatenating the two copies is that the base letters of the second half of the “doubled” string count as more significant than the accents and case of the resulting first half of the “doubled” string. This International Standard has no mechanism for handling this in a better way, where the “original” (the second half of the “doubled” string) would count as less significant than the *entire* first half of the “doubled” string. This may be handled better by having the original and copy in different ‘fields’, and construct the collation key by combining the full keys for each ‘field’. Such processing is beyond the scope of this International Standard, however.

Note that the string after preparation is used only for the collation key construction. The original string is not intended to be retrievable from the modified string, though this is possible with this way of attaining determinacy. The strings to be presented to the user are the original, by preparation untouched, strings.

Maintenance of determinacy when some of the original strings to be collated are identical, is out of the scope of this International Standard. A collation processor should, however, document if it is ‘stable’ (maintaining initial relative order of identical strings) or not. This is useful to know when collating on one field of multi-field data.

C.3 Thai string ordering – a case involving special preparation

Thai Ordering Principle

The widely accepted standard for Thai lexicographical ordering is defined in the Royal Institute Dictionary 2525 B.E. Edition (1982 A.D.), the official standard Thai dictionary. The ordering principles are:

Words are ordered alphabetically, not phonetically. Consonants order is:

ก ข ฃ ค ฅ ฆ ง จ ฉ ช ซ ฌ ญ ฎ ฏ ฐ ฑ ฒ ณ ด

ต ถ ท ธ น

บ ป ผ ฝ พ ฟ ภ ม ย ร ฤ ฤๅ ล ฦ ฦๅ ว ศ ษ ส

ห ฬ อ ฮ

(ฤ ฤๅ ฦ ฦๅ are vowels and ligatures, but put in the order according to the sounds they represent.)

Vowels are also ordered by written forms, not by sounds. Vowels order is:

ะ ั า ำ ิ ี ึ ื ุ ู เ แ โ ใ ไ

(อ ว ย are always ordered as consonants, although they sometimes act as vowels.)

Consonants always precede vowels. String comparison is performed from left to right, considering initial consonants before vowels in the same syllable.

Tones and diacritics are normally ignored, unless all other parts are equal, in which case the order is:

-็ -่ -้ -๊ -๋

Here is an ordering example:

ก ก	ก ะ เ ก ณ ฑ ์	ก ี ่
ก ร ร ม	ก ั ก	ก ึ ๋ น
ก ร ร ม ์	ก ้ า ว	ก ุ น
- ก ร ะ แ ย ่ ง	ก ำ	ก ู ด
ก ร า บ	ก ิ น	เ ก ้ ง

เ ก ล ้ า
เ ก ล ี ย ว
เ ก ้ า
เ ก า ะ
เ ก ี ่ ย ว
เ ก ี ๊ ย ะ
เ ก ื อ ก
แ ก ง
แ ก ะ
โ ก น
โ ก ร ๋ น
ใ ก ล ้
ไ ก ่
ไ ก ล
ข ้ น
ข น า บ
ข ้ า ง
ข ้ า ง ๆ
ข ้ า ง ก ร ะ
ด า น
ข ้ า ง ข ึ ้
น
ข ้ า ง ค ว า
ย

ข ้ า ง ๆ
ค ู ๆ
ข ้ า ง เ ง ิ
น
ข ้ า ง อ อ ก
เ ข น
เ ข ็ น
เ ข ่ น
เ ข ็ ด
แ ข ็ ง
แ ข ่ ง
แ ข ้ ง
แ ข ้ ง ข ว า
แ ข ็ ง ข ั น
แ ข ่ ง ข ั น
แ ข น
ค ร ร ภ -
ค ร ร ภ ์
จ ุ ม พ ล
จ ุ ํ พ ล
ช า ย
เ ฒ ่ า
เ ณ ร
ต ล า ด

ท ู ล เ ก ล ้
า
ท ู ล เ ก ล ้
า ฯ
ท ู ล เ ก ล ้
า ท ู ล ก ร ะ
ห ม ่ อ ม
น ้ า
น ้ ำ
น ี ้
บ ุ ญ ห ล ง
บ ุ ญ - ห ล ง
ป า
ป ่ า
ป ้ า
ป ๊ า
ป ๋ า
ป า น
ผ ั ด
ฯ พ ณ ฯ
พ ณ ิ ช ย ์
ย ่ อ ง
ร อ ง
ฤ ท ธ ิ ์
ฤ ษ ี

ฤ ๅ ษ ี

ล ล ิ ต า

ฦ ๅ ช า

ว ก

ศ า ล

ห ร ิ ภ ุ ญช

ั ย

ห ฤ ท ั ย

ห ล ง

แ ห ง ่

แ ห ่ ง

แ ห น ม

แ ห น ห ว ง

แ ห บ

แ ห ม

อ า น

ฮ า

Algorithmic Aspect

The above principle, with appropriate character code assignment such as TIS-620 and ISO/IEC 10646, almost allows C standard library function **strcmp()** to collate Thai strings without much more complication, except:

Leading vowels (เ - แ - โ - ใ - ไ -, corresponding to characters U0E40-U0E44), which are written before consonants, must be considered after the initial consonant. Therefore, the rearrangement is needed before comparison.

Diacritics and tone marks (-็ -ฺ -์ -๎ -่ -้ -๊ -๋) must be ignored in the first pass, and be considered at later pass if the first pass yields equality.

And these are the only two mandatory requirements for Thai string collation algorithms. No syllable structure nor word boundary analysis is required, as Thai lexicons are ordered alphabetically, not phonetically.

Leading Vowel Rearrangement

To fullfill this requirement, either a preprocessing or collating-element grouping is required. The preprocessor scans the string once and swaps every leading vowel with its succeeding letter. The preprocessed string is then passed to the normal weight calculation process. Another way to manage this is by means of collating-element formation. Every possible pair of leading vowel and consonant is defined as a collating-element, whose weight equals to that of the rearranged substring.
Note that the rearrangement of a leading vowel is simply performed with its immediate succeeding consonant. No consonant cluster analysis is needed. Indeed, doing so would result in ambiguities or yield a different order than that specified in the Royal Institute Dictionary. For example:

1. Ambiguities. The problem with ambiguity is illustrated by the word “เ พ ล า ”. It has two potential pronunciations: either as a two-syllable word, “phe-la” (meaning “time”), or as a one-syllable word, “phlao” (meaning “axle” or “abate”). A rearrangement algorithm which follows the distinct pronunciation of the potential cluster ‘พ ล ’ in this string would result in distinct keys, “พ เ ล า ” and “พ ล เ า ”, and therefore different weights, which are equally legal. Both words need to have the same weight to be sortable, however.

2. Non-conforming Ordering. To illustrate the difference in ordering caused by the treatment of consonant clusters, consider these words, shown in conforming order: “เ พ ล , เ พ ล ง , เ พ ศ ”. The correct rearrangement ignores any clusters and results in the following: “พ เ ล , พ เ ล ง , พ เ ศ ”, which sorts in the order shown.

If, however, pairs of consonants that form legal clusters were grouped as single collation elements (regardless of actual pronunciation where the potential pronunciation is ambiguous), then the results of rearrangement would be "<พ ล >เ , <พ ล >เ ง , พ เ ศ ", which would yield the (non-conforming) ordering "เ พ ศ , เ พ ล , เ พ ล ง ".

Again, if actual clusters were grouped as single collation elements (with some disambiguation effort), then the results of rearrangement would be

"พ เ ล , <พ ล >เ ง , พ เ ศ ", which would yield the (non-conforming) ordering

"เ พ ล , เ พ ศ , เ พ ล ง ".

The Multiple Levels of Character Weights

The second requirement of the algorithm, relating to the treatment of diacritics and tone marks, implies multiple levels of weights. Tone marks and diacritics must be ignored in the first level, and weigh more than consonants and vowels in the second level.

There are ten Thai decimal digits (๐ ๑ ๒ ๓ ๔ ๕ ๖ ๗ ๘ ๙), each semantically equivalent to Arabic digit 0-9, respectively. Their weights are then equal to their corresponding Arabic digits in the first level, and are different in the second level, to distinguish languages.

When punctuation marks (ฯ ๆ ฿ ๏ ๚ ๛) are concerned, another level of weights is required for them. This corresponds to the fourth level in the Common Template Table. In string ordering, punctuation marks are less significant than any tone marks and diacritics, and must be ignored in all the first three levels.

For example, "ข ้ า ง ๆ , ข ้ า ง ก บ , ข ้ า ง ๆ ค ู ๆ , ข ้ า ง จ ั น " is a valid order in the Royal Institute Dictionary. In the first level, the considered weights are ข า ง , ข า ง ก บ , ข า ง ค ู , ข า ง จ ั น respectively.

The third level is not defined for Thai string ordering, but is reserved for tailoring.

Annex D -- Tutorial on solutions brought by this standard to problems of lexical ordering (informative)

Why aren't existing standard codes, character by character comparisons, and commercial sort programs appropriate for sorting, and what must be done to solve the problem? For clarity, this discussion will start with the Latin script.

i. Sorting, in any language using the Latin script, including English, using standard ISO/IEC 646 coding, does not follow traditional dictionary sequence, which is the minimum the average user needs.

Example: Sorting the list "august", "August", "container", "coop","co-op", "Vice-president", "Vice versa" gives the following order, if ISO/IEC 646 coding is used and a simple sort following binary order is performed:

August
Vice versa
Vice-president
august
co-op
container
coop

This ordering is obviously incorrect.

ii. Transforming lower case to upper case and removing special characters yields a sorted list acceptable to users, but also yields unpredictable results.

Example: Sorting the list "August", "august", "coop", "co-op" gives the following order:

August
august
coop
co-op

Sorting the same list with a different initial order, say, "august", "co-op", "August", "coop" may give a different order with this method:

august
August
co-op
coop

iii. If accented characters are introduced using for example any ISO/IEC 8-bit character set, the same problems encountered in examples i and ii above are amplified but they share the same causes.

iv. If tables are reorganized to make all related characters contiguous, one might think that a simplified single-character sort would result, but this does not work either. Take upper and lower case unaccented letters as an example. If code position 01 is assigned to "a", code position 02 assigned to "A", code position 03 to "b", code position 04 to "B" and so on, a list sorted directly according to these rearranged values will yield the following:

Sorted List	Internal Values
aaaa	01010101
abbb	01030303
Aaaa	02010101
Abbb	02030303

This is also predictable, but remains obviously incorrect for any country with regard to cultural expectations.

v. The only solution is to decompose the initial data in a way which will respect traditional lexical order, and at the same time ensure absolute predictability. For the Latin script, this necessitates at least four levels:

1. The first decomposition renders information to be sorted case-insensitive and insensitive to diacritical marks, removing all special characters (which have no pre-established traditional order.

An example using English:

"résumé" (’curriculum vitae’) becomes "resume" (‘begin again’), without any accent.

An example using French:

"Vice-légation" becomes "vicelegation", with no accent, no upper case and no hyphen.

An example using German:

"groß" becomes "gross", with the sharp-s being converted to double-s to render it case insensitive.

In some languages including Spanish or Nordic languages, some extra letters are added to the 26 fixed letters of the English, French and German alphabets, which are not ordered according to the expectations of those languages. This demonstrates the need for adaptability.

2. The second decomposition breaks ties on quasi-homographs, that is, strings that differ only because they have different diacritical marks. In English, "resumé" and "résumé" are quasi-homographs. Traditional English lexical order requires that "resume" always comes before "résumé" (which sorting using only the first level would not guarantee). In this case, the tradition does not explicitly specify whether "resumé" should come before "résumé", though this would seem logical: most English and German dictionaries only state that unaccented words precede the accented words - However German dictionary generally employ the German standard DIN 5007, which states more precise rules.

Here another characteristic is introduced. In French, because of the large number of multiple quasi-homograph groups formed of more than 2 instances, the most important dictionaries follow the following rule: accents are generally not taken into account for sorting, but in case of homographic ties, the *last difference* in the word determines the correct order between two given words, a priority order being then assigned to each type of accent. According to this, "coté" should be sorted after "côte" but before "côté". This is easy to implement with "backwards" tailoring: a number is assigned to each character of the data to be sorted, representing either a letter with an accent or a letter with no accent at all, but these numbers are stacked instead of being added to a linear list: in other words, the resulting string is made starting from the last character of the original data and processing in a backwards direction.

Example: to obtain an order respecting this rule: "cote, "côte", "coté", "côté", numbers could be assigned indicating respectively "****", "**c*", "a***", "a*c*", where "*" means no accent, "a" means acute accent, "c" circumflex accent. Here this scheme is sufficient to break the tie correctly at this second level.

3. The third decomposition breaks ties for quasi-homographs which differ only because upper-case and lower-case characters are used. This time, the tradition is well established in German dictionaries, where lower case always precedes upper case in homographs, while the tradition is not well established in French dictionaries, which generally use only accented capital letters for common word entries. In known French dictionaries where upper and lower case letters are mixed, the capitals generally come first, though this is not an established and stated rule, because there are numerous exceptions. English has no monolithic practice for this, a bit like French. So for a Common Template it is advisable to use the well-established German tradition, if one wants to group the largest possible number of languages together without affecting others. Note that in Denmark, upper case is specified to precede lower case, a different but well-established rule. This is a second fact which demonstrates the need for adaptability in the model used in this International Standard.

Example: to have the following order: "august", "August", numbers could be assigned indicating respectively "llllll", "ulllll", where "l" means lower case and "u" upper case.

4. The fourth decomposition breaks the final tie which, in general, does not correspond to any strong tradition, namely, the tie between quasi-homographs differing only because they contain special characters. Breaking this tie is essential to ensure the absolute predictability of ordering as well as enabling the ordering of strings composed only of special characters. Since the traces of special characters were removed from the original data to form the three first orders of decomposition, simply putting them sequentially in the fourth order of decomposition would mean that their position would be lost. These positions are quite important to solve remaining ties and in consequence the original positions of these special characters must be retained: two quasi-homographs could each contain a common special character in different positions and thus be strictly different (example: "ab*cd" is different from "a*bcd" despite they share one and only one common special character).

Example: to obtain the following order: "coop", "co-op", "coop-", numbers could be assigned respectively according to the following pattern: "d", "d3-" and "d5-", where "d" is an ever-present delimiter separating this decomposition from the first three in case all four decompositions are to be concatenated to form a single sorting key based on numeric values (see discussion in the next paragraph). "3-" means a hyphen in position 3 of the original string. "5-" means a hyphen in position 5, and so on.

These four decompositions can be structured using a four-level key, concatenating the subkeys from the highest significance to the lowest. If the coded assignment of numbers is done properly, instead of necessitating a cumbersome exception process for dealing with homographs, all decompositions may be made at once and resulting strings concatenated and passed through a standard ordering program sorting in numeric order. To attain this result, it is sufficient that the numbers chosen for the first decomposition code set be greater than numbers chosen for the second one, the second one's greater than the third one's, and that the delimiter chosen for the fourth decomposition be less than the lowest possible number coded elsewhere for the sort (a delimiter called logical zero), in which case no restriction applies to the content of the fourth decomposition. An easier implementation might just choose to put the lowest value possible as a delimiter between each subkey, in which case no restriction ever applies.

This method was fully described with tables in *Règles du classement alphabétique en langue française et procédure informatisée pour le tri*, Alain LaBonté, Ministère des Communications du Québec, 1988-08-19, ISBN 2-550-19046-7.

Reduction techniques have been designed to considerably shorten space requirements. As no implementation is required to use specific numbers for weights and neither reduction nor compression is required, this issue is outside the scope of this International Standard. Nevertheless, it is interesting to note that implementation can be optimized. This has been improved over time and is easy to accomplish, some methods being more efficient than others.

A public-domain reduction technique is described in details (with numerous examples) in *Technique de réduction - Tris informatiques à quatre clés,* Alain LaBonté, Ministère des Communications du Québec, 1989-06 (ISBN 2-550-19965-0).

vi. For a number of languages, the Common Template presented in this standard will need to be adapted, both in the table values for the four orders of keys (which can require redefining characters or introducing multicharacter collating elements into the table) and in the potential context analysis processing necessary to achieve culturally correct results for users of these languages. To illustrate this (without discussing context analysis which is not necessary in what follows), examples of dictionary sequences are given here for two languages which native order is not in the Common Template table:

Traditional Spanish (where "ch" is greater than "cu" and "ña" is greater than "no"):
cuneo<cúneo<chapeo<nodo<ñaco

(Comparative French/English/German sort:
chapeo<cuneo<cúneo<ñaco<nodo)

Danish (where "a" is less than "c", "cz" is less than "cæ" and "cø", and "aa" is equivalent to "å", which is greater than "z", even in cases where it is pronounced differently):
Alzheimer<czar<cæsium<cølibat< Aachen<Aalborg<Århus

(Comparative French/English/German sort:
Aachen<Aalborg< Alzheimer<Århus<cæsium<cølibat<czar)

Annex E – BIBLIOGRAPHY (informative)

The following standards and documents are considered relevant to this standard, in addition to the normative references.

CAN/CSA Z243.4.1-1998 – *Canadian Alphanumeric Ordering Standard – A National Standard of Canada*, Canadian Standards Association

CAN/CSA Z243.230-1998 – Minimum Canadian Software Localisation Conventions – A National Standard of Canada

DS 377 (1980) – DS 377:1980 *Alfabetiseringsregler* – Dansk Standard

ISO/IEC 646, *Information technology – ISO 7-bit coded character set for information interchange*

ISO/IEC 2022, *Information technology – Code extension techniques*

ISO/IEC 6937, *Information technology – Coded character sets for text communication*

ISO/IEC 8859-1, *Information technolog – 8-bit single-byte coded graphic character sets -- Part 1: Latin alphabet No. 1*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets -- Part 15: Latin alphabet No. 9*

ISO/IEC 9945-2, *Information Technology – Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities*
ISO/IEC DTR 14652, *Information Technology -- Specification Method for Cultural Conventions*

Règles du classement alphabétique en langue française et procédure informatisée pour le tri, Conseil du trésor du Québec – URL: http://www.tresor.gouv.qc.ca/doc/classm.htm

Retskrivningsordbogen – 2nd edition 1996, Dansk Sprognævn & Aschehoug Dansk Forlag A/S

Technique de réduction - Tris informatiques à quatre clés, Conseil du trésor du Québec – URL: http://www.tresor.gouv.qc.ca/doc/techtri.htm

The Unicode Standard, Version 2.0, The Unicode Consortium, Addison Wesley Developers Press, ISBN 0-201-48345-9

Unicode Technical Report no. 8, The Unicode Standard, Version 2.1, The Unicode Consortium – URL: http://www.unicode.org/unicode/reports/tr8/

Unicode Technical Report no. 10, Unicode Collation Algorithm, The Unicode Consortium – URL: http://www.unicode.org/unicode/reports/tr10/

Unicode Technical Report no. 15, Unicode Normalization Forms, The Unicode Consortium – URL: http://www.unicode.org/unicode/reports/tr15/

END OF THIS INTERNATIONAL STANDARD