ICFP 2001 Programming Contest
Challenge Task

version 6

Damien Doligez, Luc Maranget, Pierre Weis

July 28, 2001

1 The SML/NG markup language

The W4C (World Wide Wireless Web Consortium) has just published the specification of SML/NG (Simple Markup Language – New Generation), a simplified version of XXHTML designed for the new generation of hypertext rendering micro-devices, running on hardware with reduced computational capacity such as wristtop computers, thumbnail-worn PDAs, and internet-enabled ice boxes.

The programming task is to design and implement an optimiser for SML/NG that will simplify the source documents and reduce their size.

2 Description of SML/NG

A document is composed of text and tags. A tag is a sequence of characters (the tag's *name*) between `<` and `>`. Anything else in a document is called *text*. For example,

```
foo<b>bar</b>
```

is the text `foo` followed by tag `<b>`, text `bar`, and tag `</b>`.

The characters `<` and `>` can only appear in a document as part of a tag (but of course the strings `&lt;` and `&gt;` can appear in the text).

2.1 Tags

A tag whose name begins with the character `/` is a closing tag. The corresponding open tag has the same name without the leading `/`. For instance, `</b>` is the closing tag corresponding to the open tag `<b>`.

All tags in a document appear in pairs, composed of an open tag and the corresponding closing tag (in this order). The region of the document between two matching tags (including the tags) is called the *range* of this pair. For example, in `foo<b>bar</b>`, the range of the `<b>`, `</b>` pair is `<b>bar</b>`.

Tags are properly nested: for any two ranges, either they are disjoint or one is entirely included in the other.

Each tag changes the attributes of text within its range as described below:

`B` (bold) set the B attribute

`EM` (emphasis) invert the EM attribute

`I` (italic) set the I attribute

`PL` (plain) reset the U level to 0 and unset the B, EM, I, S, and TT attributes

`S` (strong emphasis) set the S attribute

`TT` (typewriter) set the TT attribute

`U` (underline) increment the U level (but not above 3)

`0...9` (size) set the size to 0...9

`r` (color) set the color to red

`g` set the color to green

`b` set the color to blue

`c` set the color to cyan

`m` set the color to magenta

`y` set the color to yellow

`k` set the color to black

`w` set the color to white

Note: case *is* significant in tag names.

There is one additional interaction between the attributes: S hides the EM state (i.e. where the S attribute is set, the EM attribute is irrelevant).

2.2 White space

There are 4 non-printing characters: SPC (ASCII code 0x20), CR (ASCII code 0x0D), LF (ASCII code 0x0A), and TAB (ASCII code 0x09). Any sequence of these characters is called white space and is equivalent to one SPC character, except where the TT attribute is set (there are other conditions that prevent spaces from being collapsed, see section 3). In the parts of the document where TT is set, the whitespace characters are significant and must be preserved exactly. A document contains only these four characters and printable ASCII characters (codes 0x21 to 0x7E, included).

There is another interaction between whitespace and flags: the EM, I, B, and S attributes are irrelevant for whitespace; moreover, color is irrelevant where U = 0.

2.3 BNF grammar

The BNF grammar of documents is given below.

document ::= *document document*
| *textchar* *
| <B> *document* </B>
| <EM> *document* </EM>
| <I> *document* </I>
| <PL> *document* </PL>
| <S> *document* </S>
| <TT> *document* </TT>
| <U> *document* </U>
| <0> *document* </0>
| <1> *document* </1>
| <2> *document* </2>
| <3> *document* </3>
| <4> *document* </4>
| <5> *document* </5>
| <6> *document* </6>
| <7> *document* </7>
| <8> *document* </8>
| <9> *document* </9>
| <r> *document* </r>
| <g> *document* </g>
| *document*
| <c> *document* </c>
| <m> *document* </m>
| <y> *document* </y>
| <k> *document* </k>
| <w> *document* </w>

textchar ::= any printable character except < and >
| CR | LF | TAB | SPC

3 Specification

This section defines the *meaning* of a document. Two documents are said to be equivalent if they have the same meaning.

The meaning of a document is a sequence of decorated characters; a decorated character is a character associated with a property record. A property record has 8 fields:

b a boolean

em a boolean

i a boolean

s a boolean

tt a boolean

u an integer between 0 and 3 (included)

size an integer between 0 and 9 (included)

color an element of $\{r, g, b, c, m, y, k, w\}$

To compute the meaning of a document given a current context (a context is a property record), consider the document as a sequence of characters and tags. Treat each element of this sequence in turn as follows:

- if it is a printing character, output it, decorated by the current context
- if it is a whitespace character, compute the current space context, which is the current context modified in the following way: the **s**, **em**, **i**, and **b** flags are unset, and if **u** is 0 then **color** is set to w. If the **tt** flag is true in the current context, then output the input character decorated by the current space context; else if the previous output character was a SPC decorated with the same space context then do nothing; otherwise, output a SPC decorated with the current space context.
- if it is an open tag, save the current context and change it according to the tag name as follows:

  `B` set the **b** flag

  `EM` invert the **em** flag if the **s** flag is not set; otherwise do nothing

  `I` set the **i** flag

  `PL` unset the **s**, **em**, **i**, **b**, and **tt** flags, and set **u** = 0.

  `S` set the **s** flag and unset the **em** flag

  `TT` set the **tt** flag

  `U` if **u** is less than 3, increment it; otherwise do nothing

  `0...9` set **size** accordingly

  `r, g, b, etc.` set **color** accordingly

- if it is a closing tag, restore the context that was saved at the corresponding open tag

The meaning of the document is the sequence of decorated characters output by the above algorithm.

A root context is any context with **u** = 0 and **b** = **em** = **i** = **s** = **tt** = false (**size** and **color** can have any value). Two documents are *equivalent* if they have the same meaning in every possible root context (i.e. for all values of **size** and **color**).

You can use the on-line document checker and equivalence tester at `http://cristal.inria.fr/ICFP2001/prog-contest/validator.html`

Examples

For example, the following pairs of documents are equivalent:

```
<r>  xxx </r><b> yyy  </b>
<r> xxx  <b>  yyy </b></r>

<EM> xxx <EM> yyy </EM> zzz </EM>
<EM> xxx</EM> yyy <EM>  zzz </EM>

<B> xxx <B> yyy </B></B>
<B> xxx  yyy </B>

<r> xxx </r><b> </b><r> yyy </r>
<r> xxx yyy </r>

<EM> xxx <S> yyy </S></EM>
<EM> xxx </EM><S> yyy </S>

<I> xxx </I> yyy <I> zzz </I>
<I> xxx <PL> yyy </PL> zzz </I>
```

The following pairs of documents are not equivalent:

```
<TT><r>  xxx </r><b> yyy  </b></TT>
<TT><r> xxx  <b>  yyy </b></r></TT>
```

Reason: multiple spaces are significant within `TT`.

```
<B> xxx <I> yyy </I> zzz </B>
<B> xxx </B><I> yyy </I><B> zzz </B>
```

Reason: `yyy` is both in italics and in bold in the first document but only in italics in the second one.

```
<U> xxx <U> yyy </U></U>
<U> xxx  yyy </U>
```

Reason: `yyy` is underlined twice in the first document but only once in the second one.

```
<U><r> xxx </r><b> </b><r> yyy </r></U>
<U><r> xxx   yyy </r></U>
```

Reason: the first document has three underlined spaces between `xxx` and `yyy` because the middle one is in blue; the second document has only one red underlined space at that point.

4 The task

You must write a program to optimise SML/NG documents. Your program will be given a correct SML/NG document on its standard input, and it must output (on stdout) an equivalent document that is as small as possible. The size of a document is simply defined as its length in bytes.

For example, opportunities for optimisation include the following:

- **whitespace compression**
  replacing a white space sequence with a single space or newline
- **redundancy elimination**
  changing `<B><I><B>foo</B></I></B>`
  into `<I><B>foo</B></I>`
- **overlap inversion**
  changing `<B><I>bla bla</I></B><TT><I><B>foo bar</B></I> truc</TT>`
  into `<B><I>bla bla<TT>foo bar</TT></I></B><TT> truc</TT>`
- **PL shortcut**
  changing `<I><S><TT>foo</TT></S></I> bar <TT><S><I> gee</I></S></TT>`
  into `<I><S><TT>foo<PL> bar </PL> gee</TT></S></I>`
- **whitespace simplification**
  changing `<B><I><r>bla bla </r> </I> </B> <r><I> barfoo</I></r>`
  into `<I><r><B>bla bla </B>barfoo</r></I>`
- **EM elimination**
  changing `<S> foo <EM> bar </EM> </S> <EM> <EM> foo </EM> </EM>.`
  into `<S> foo bar</S> foo .`
- **color nesting**
  changing `<r>aaa</r><g>bbb</g><b>ccc</b><g>ddd</g><r>eee</r>`
  into `<r>aaa<g>bbb<b>ccc</b>ddd</g>eee</r>`

There will also be a limitation on the amount of time that your program may use to do its work. The time limit will depend on the input file and it will be given to your program as a number of seconds, both in its first command-line argument and in the `TLIMIT` environment variable. The limit will never be less that 180 (i.e. 3 minutes).

Note that the limit is real time (a.k.a wall-clock time), not CPU time.

5 Judgement criteria

Your programs will be ranked according to the following criteria:

1. Correctness. Every program that crashes or gives the wrong result (i.e. some output that is not equivalent to the input) on any one of our test inputs will be mercilessly eliminated from the competition.
2. Stupidity. Every program that gives a result bigger than the input on any of the inputs will also be eliminated.

3. Output size. The remaining programs will be ranked according to the size of their outputs on a well-chosen set of inputs. The inputs will be generated using a variety of techniques such as hand-writing, translation from HTML, random generation. These inputs can be big (up to five megabytes).

4. Speed of optimisation. If the top programs are very close to each other using the previous criterion, we will use speed as a tie breaker.

6 Online stuff

The following Web pages may be of interest to you:

- Contest home page: `http://cristal.inria.fr/ICFP2001/prog-contest/`
- FAQ: `http://cristal.inria.fr/ICFP2001/prog-contest/faq.html`
- News: `http://cristal.inria.fr/ICFP2001/prog-contest/news.html`
- Document checker and equivalence tester: `http://cristal.inria.fr/ICFP2001/prog-contest/validator.html`
- Procedure for submitting entries: `http://cristal.inria.fr/ICFP2001/prog-contest/procedure.html`

7 Good luck

Have fun !