1.   Function and Use.

This small program will convert Big 5 encoding with CNS encoded Chinese characters using the *Chinese Encoding Framework (CEF)* into a 'preprocessed' form. The need of this program arises from the fact that Big 5 encoding uses the characters '\', '{', and '}' which have special meanings in TeX.

Use this program as a filter:

```
cef5conv < input_file > output_file
```

2. The program.

In contrast to `cefconv` two tasks will be executed:

Replacing all occurrences of Big 5 encoded characters `XY` (`X` and `Y` are the first and the second byte of the character) with `^^7fX^^7fZZZ^^7f`, where `ZZZ` represents the second byte as a decimal number. `0x7F` is used as a control character and a delimiter.

Replacing CEF macros of the form `&xx-yyzz;` (`xx` can be C1–C7 for the CNS planes 1–7, C0 for Big 5 encoding, an encoding CX reserved for IRIZ, a private encoding CY, and U for Unicode encoding; `yyzz` is a hexadecimal representation of the code point in this plane) with

```
^^7f72^^7fXX^^7f^^7f"0yy^^7f"0zz^^7f   .
```

`XX` is the corresponding CJK encoding of `xx`; the number '72' specifies a macro in the file `MULEenc.sty` which further processes this representation – it is automatically loaded by the `CJK` package.

Additionally we define a TeX macro at the very beginning to signal a preprocessed file.

The following code is very simple. No error detection is done because TeX which will see the output of `cef5conv` complains loudly if something is wrong.

#define *banner* `"cef5conv␣(CJK␣ver.␣4.8.1)"`

#include `<ctype.h>`
#include `<stdio.h>`
#include `<stdlib.h>`

```
  int  main(argc, argv)
      int argc;
      char *argv[ ];
  {int ch, i;
   unsigned char in[16];
   unsigned char out[32];
   unsigned char *inp, *outp;

   fprintf(stdout, "\\def\\CNSpreproc{%s}", banner);

   ch = fgetc(stdin);

   while (! feof(stdin))
     {if (ch ≥ #A1 ∧ ch ≤ #FE)
       {fprintf(stdout, "\177%c\177", ch);

         ch = fgetc(stdin);
         if (! feof(stdin))
           fprintf(stdout, "%d\177", ch);
       }
     else if (ch ≡ '&')                         /∗ the macro test is hardcoded to make things simple ∗/
       {inp = in;
        outp = out;
        *inp = ch;
        *(++inp) = fgetc(stdin);
```

if $(*inp \equiv \text{'C'} \wedge \,! feof\,(stdin\,))$
  $\{*(+\!\!+inp) = fgetc\,(stdin\,);$
   if $(*inp \equiv \text{'0'} \wedge \,! feof\,(stdin\,))$
    $\{*(outp+\!\!+) = \text{'B'};$
     $*(outp+\!\!+) = \text{'g'};$
     $*(outp+\!\!+) = \text{'5'};$
    $\}$
   else if $(*inp \geq \text{'1'} \wedge *inp \leq \text{'7'} \wedge \,! feof\,(stdin\,))$
    $\{*(outp+\!\!+) = \text{'C'};$
     $*(outp+\!\!+) = \text{'N'};$
     $*(outp+\!\!+) = \text{'S'};$
     $*(outp+\!\!+) = *inp;$
    $\}$
   else if $((*inp \equiv \text{'X'} \vee *inp \equiv \text{'Y'}) \wedge \,! feof\,(stdin\,))$
    $\{*(outp+\!\!+) = \text{'C'};$
     $*(outp+\!\!+) = \text{'E'};$
     $*(outp+\!\!+) = \text{'F'};$
     $*(outp+\!\!+) = *inp;$
    $\}$
   else
     goto $no\_macro;$
  $\}$
else if $(*inp \equiv \text{'U'} \wedge \,! feof\,(stdin\,))$
  $\{*(outp+\!\!+) = \text{'U'};$
   $*(outp+\!\!+) = \text{'T'};$
   $*(outp+\!\!+) = \text{'F'};$
   $*(outp+\!\!+) = \text{'8'};$
  $\}$
else
  goto $no\_macro;$

$*(+\!\!+inp) = fgetc\,(stdin\,);$
if $(*inp \neq \text{'-'} \vee feof\,(stdin\,))$
  goto $no\_macro;$

$*(outp+\!\!+) = \text{'}\backslash\text{177'};$
$*(outp+\!\!+) = \text{'}\backslash\text{"'};$
$*(outp+\!\!+) = \text{'0'};$

$*(+\!\!+inp) = fgetc\,(stdin\,);$
if $(isxdigit\,(*inp) \wedge *inp < {}^{\#}80 \wedge \,! feof\,(stdin\,))$
  $*(outp+\!\!+) = toupper\,(*inp);$
else
  goto $no\_macro;$

$*(+\!\!+inp) = fgetc\,(stdin\,);$
if $(isxdigit\,(*inp) \wedge *inp < {}^{\#}80 \wedge \,! feof\,(stdin\,))$
  $*(outp+\!\!+) = toupper\,(*inp);$
else
  goto $no\_macro;$

$*(outp+\!\!+) = \text{'}\backslash\text{177'};$
$*(outp+\!\!+) = \text{'}\backslash\text{177'};$
$*(outp+\!\!+) = \text{'}\backslash\text{"'};$
$*(outp+\!\!+) = \text{'0'};$

```
    *(++inp) = fgetc(stdin);
    if (isxdigit(*inp) ∧ *inp < #80 ∧ !feof(stdin))
      *(outp++) = toupper(*inp);
    else
      goto no_macro;

    *(++inp) = fgetc(stdin);
    if (isxdigit(*inp) ∧ *inp < #80 ∧ !feof(stdin))
      *(outp++) = toupper(*inp);
    else
      goto no_macro;

    *(outp++) = '\177';
    *outp = '\0';

    *(++inp) = fgetc(stdin);
    if (*inp ≠ ';' ∨ feof(stdin))
      goto no_macro;

    outp = out;
    fprintf(stdout, "\17772\177");
    while (*outp)
      fputc(*(outp++), stdout);

    ch = fgetc(stdin);
    continue;
  no_macro:
    ch = *inp;
    i = inp − in;
    inp = in;
    while (i−−)
      fputc(*(inp++), stdout);
    continue;
    }
  else
    fputc(ch, stdout);

  ch = fgetc(stdin);
  }
exit(EXIT_SUCCESS);
return 0;                                                          /* never reached */
}
```