1. Function and Use.

This small program will convert SJIS 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:

```
cefsconv < input_file > output_file
```

2. The program.

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

Replacing all occurrences of two byte SJIS encoded characters `XY` with `XZZZ^^7f` (`X` and `Y` are the first and the second byte of the character; `ZZZ` represents the second byte as a decimal number).

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 necessary to explicitly load this file with `\usepackage`.

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 `cefsconv` complains loudly if something is wrong.

```
#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{}");

    ch = fgetc(stdin);

    while (!feof(stdin))
    {if ((ch ≥ #81 ∧ ch ≤ #9F) ∨ (ch ≥ #E0 ∧ ch ≤ #EF))
      {fputc(ch, stdout);

        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 ≡ 'C' ∧ !feof(stdin))
         {*(++inp) = fgetc(stdin);
          if (*inp ≡ '0' ∧ !feof(stdin))
           {*(outp++) = 'B';
            *(outp++) = 'g';
            *(outp++) = '5';
           }
          else if (*inp ≥ '1' ∧ *inp ≤ '7' ∧ !feof(stdin))
           {*(outp++) = 'C';
```

```
        *(outp++) = 'N';
        *(outp++) = 'S';
        *(outp++) = *inp;
      }
      else if ((*inp ≡ 'X' ∨ *inp ≡ 'Y') ∧ !feof(stdin))
      {*(outp++) = 'C';
        *(outp++) = 'E';
        *(outp++) = 'F';
        *(outp++) = *inp;
      }
      else
        goto no_macro;
    }
    else if (*inp ≡ 'U' ∧ !feof(stdin))
    {*(outp++) = 'U';
      *(outp++) = 'T';
      *(outp++) = 'F';
      *(outp++) = '8';
    }
    else
      goto no_macro;
    *(++inp) = fgetc(stdin);
    if (*inp ≠ '-' ∨ feof(stdin))
      goto no_macro;
    *(outp++) = '\177';
    *(outp++) = '\"';
    *(outp++) = '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++) = '\177';
    *(outp++) = '\"';
    *(outp++) = '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 */
}
```