CS520

PROGRAM 1

Description

Write two C programs:

- **utf8to16**.

  The program reads a file containing UTF-8 encodings of a sequence of Unicode characters, decodes the UTF-8 to determine the Unicode characters, and writes those decoded characters in UTF-16 format to an output file.

  The program will take two arguments: the name of the input file as the first argument and the name of the output file as the second argument. Print an appropriate error message if the input file cannot be opened for reading or the output file cannot be opened for writing. (If the user specifies the same file for both input and output, then the behavior is undefined, meaning that there is no particular requirement for how you handle this case.)

  The input file will start with a BOM. However, the BOM is not needed to decode the file, since UTF-8 files are simply a sequence of bytes. If there is no BOM, report the error and terminate the program.

  The output file should be written in Big Endian format. Since the input file will start with a BOM, the output file will also start with a BOM.

  A file that contains only a BOM is okay. Simply write an output file that contains only a BOM.

  A file is a valid UTF-8 file if it does **not** contain any of these errors:

  - An overlong encoding.
  - An unexpected continuation byte.
  - A start byte not followed by enough continuation bytes.
  - A 4-byte sequence starting with 0xF4 that decodes to a value greater than 0x10FFFF.
  - A sequence that decodes to a *noncharacter*.
  - A sequence that decodes to a value in the range 0xD800-0xDFFF (i.e. a leading or trailing surrogate).

  If an error is detected, print an appropriate error message to stderr that includes the offset in the file for the start byte for the sequence that is in error. (Display the offset as a decimal number).

  You may exit the program after reporting the first error.

  The program will return a status of -1 if an error is encountered; otherwise it will return a status of 0.

  Put all your source code in the file **utf8to16.c**.

- **utf16to8**.

  The program reads a file containing UTF-16 encodings of a sequence of Unicode characters, decodes the UTF-16 to determine the Unicode characters, and writes those decoded characters in UTF-8 format to an output file.

  The program will take two arguments: the name of the input file as the first argument and the name of the output file as the second argument.

  The input file will start with a BOM to indicate whether the file is stored in Little Endian or Big Endian format. If there is no BOM, report the error and terminate the program.

  Write the BOM, encoded as UTF-8, to the output file. A file that contains only a BOM is okay. Simply write an output file that contains only a BOM.

  A file is a valid UTF-16 file if it does **not** contain:

  - A value that denotes a *noncharacter*.
  - Unpaired surrogates: A value in the range 0xD800 to 0xDBFF not followed by a value in the range 0xDC00 to 0xDFFF, or any value in the range 0xDC00 to 0xDFFF not preceded by a value in the range 0xD800 to 0xDBFF.

  If an error is detected, print an appropriate error message to **stderr** that includes the offset in the file for the start byte for the UTF-16 value that is in error. (Display the offset as a decimal number.)

  You may exit the program after reporting the first error.

  The program will return a status of -1 if an error is encountered; otherwise it will return a status of 0.

  Put all your source code in the file **utf16to8.c**.

You are not allowed to use any *iconv* library routines. The goal is for you to solve this problem yourself at a low level, in order to get experience with bit manipulation in C.

Define these constants used for error messages that you need to display if there is an error (along with the offset or bytes instead of BOM whenever relevant):

```
#define ERR_TWO_ARGUMENTS "Requires 2 arguments, input file and output file"
#define ERR_INVALID_FILE "Cannot open file:"
#define ERR_CONTINUATION_BYTE "Missing or Unexpected Continuation byte at offset:"
#define ERR_OVERLONG_ENCODING "Overlong Encoding at offset:"
#define ERR_NONCHARACTER "The sequence decodes to a noncharacter: start byte at offset:"
#define ERR_RESERVED_SURROGATES "The sequence decodes to a value reserved for surrogates: start byte at offset:"
#define ERR_OUT_OF_RANGE "The sequence decodes to a value greater than 0x10FFFF: start byte at offset:"
#define ERR_INCOMPLETE_CODE_UNIT "Incomplete UTF-16 code unit at offset:"
#define ERR_UNPAIRED_SURROGATE "Unpaired surrogate at offset:"
#define ERR_MISSING_BOM "Missing BOM" // followed by the 2 or 3 bytes that you see at the start of the file in hex (if there are at least 2 or 3 bytes)
#define ERR_INCORRECT_START_BYTE "Incorrect start byte at offset:" //default error if all other specific error cases fail
```

Examples of error messages:

Overlong Encoding at offset: 6

Missing BOM e0 a0 80