HUFFMAN CODING COMPRESSION PROJECT

I. Program Inputs:

There will be three kinds of files the program is able to read / write (depending on the mode):

- Input Files: it can either be a text file or a binary file, like a JPG, MP3, EXE, or video.
- 2) Encoded files. These will be (Huffman-compressed) files encoded by the program. Most of the time, it is a file with huf extension.
- 3) Tree-Building information files. These will contain the information required to build a Huffman tree from some (particular) input file. These will always be exactly 510 bytes. Most of the time, it is a file with htree extension.

- Note: a huf file always has the information for a htree embedded inside it – the last 510 bytes of a huf tells us how to build the tree in order to decode the rest of the file.

II. Program Operations:

The -m switch determines the mode, and will be one of:

- -me encode a file
- -md decode a file
- -mt build a tree-builder from a file
- -met encode a file with a specified tree-builder file
- -m followed by anything other than e, d, t, or et is not valid.

To specify the input file, use -i:<filename>

To specify the output file, use -o:<filename>

To specify the tree-builder input file, use -t<filename>

To ask for help, use -h or -? or -help

<u>Mode 1 – Show Help:</u>

Syntax: HUFF –h or HUFF -? or HUFF –help

Description: Displays the proper usage options to the user and exits.

<u>Mode 2 – Encode Directly from Input File:</u>

Syntax: HUFF –me -i:file1 [-o:file2]

Description: Encode file1, placing the output into file2. The program reads file1, build a Huffman tree from the file, and then produce file2, whose last 510 bytes will be the tree-building information, and the remainder of which will be the Huffman-encoded bit stream created from file1, padded to fill the last byte of the bit stream. Specifying file2 (the output file) is optional. If the user omits the output file, the program takes file1, and remove its extension, and append .huf. If file2 is omitted and file1 has no extension, then append .huf to the name of file1. If file2 is specified, use its name as-supplied. File1 and file2 must not refer to the same file. If either the input file can't be located, or the output file can't be created, display an error message and exit.

Examples:

HUFF –me -i:shakespeare.txt

⇨ Encode Shakespeare.txt → Shakespeare.huf

HUFF –me -i:hamlet.txt -o:hamlet.enc

⇨ Encode Hamlet.txt → Hamlet.enc

HUFF –me -i:hamlet

⇨ Encode Hamlet → Hamlet.huf

Mode 3 – Decode:

Syntax: HUFF –md -i:file1 -o:file2

Description: Decode Huffman-encoded file1 into file2. File1 will contain a 510-byte tree-builder block, preceded by the Huffman-encoded bit stream for some original input file (perhaps the output from –me above). The program will read in the tree-builder block, construct the tree, and use it to decode file1, placing the result into file2. File2 should be bit-identical to the original input file. **File2 is not optional**– the user must specify both file names, and again, both file names must not refer to the same file. If the input file doesn't exist, or the output file can't be created, issue an error message and exit.

Examples:

HUFF –md -i:hamlet.enc -o:hamlet2.txt

⇨ Decode hamlet.enc → hamlet2.txt

HUFF –md -i:shakespeare.huf -o:all-plays.txt

⇨ Decode Shakespeare.huf → all-plays.txt

Mode 4 – Create A Tree-Building File:

Syntax: HUFF –mt -i:file1 [-o:file2]

Description: Your program will read file1 and use the information in it to produce a 510-byte tree builder file named file2. As with the encoding mode, file2 is optional. If file2 is not specified and file1 has an extension, replace it with .htree. If file2 is not specified and file1 does not have an extension, append .htree to file1's name to produce file2's name. If file2 is specified, then use file2 as-supplied. Both file names must not refer to the same file. If the input file can't be located, or if the output file can't be created, display an error message to that effect and exit.

Examples:

HUFF –mt -i:Shakespeare

⇨ Produces 510-byte shakespeare.htree

HUFF –mt -i:shakespeare.txt -o:shakespeare.htree

⇨ Produces 510-byte shakespeare.htree

HUFF –mt -i:hamlet -o:hamlet.510

⇨ Produces 510-byte hamlet.510

<u>Mode 5 – Encoding with A Specified Tree-Builder:</u>

Syntax: HUFF –met -i:file1 -t:file2 [-o:file3]

Description: Your program will read input from file1 and rather than building a tree from the contents of file1, encode it using a tree built from the 510-byte tree-builder information in file2, producing file3 (note: file3 is optional, and will default to the same filename as file1, except with an extension of .huf, which will be either changed or appended to file1's name as appropriate). The last 510 bytes of file3 will be the tree-builder information from file2, preceded by the Huffman-encoded bit stream from file1.

Examples:

HUFF –met -i:Shakespeare.txt -t:bible.htree

⇨ Produces encoded Shakespeare.huf from Shakespeare.txt, using the treebuilder information in bible.htree

```
HUFF –met -i:hamlet.txt -t:Hunchback.htree -o:hamlet.huf
```

⇨ Produces encoded hamlet.huf from hamlet.txt, using the tree-builder information in hunchback.htree.