Goal: Create an implementation of LZ78 compression with executables for compression and decompression, and libraries for tries, word tables, and I/O buffering.

Pseudo code:

encode.c (main):

- check if infile is accessible and use open() for infile and outfile, call fstat(), call write_header, call trie_create, call read_sym to parse in symbols from infile, and { call curr_sym, set next_node to trie_step(curr_node, curr_sum), if next_node isn't null, { set prev_node to current_node, and curr_node to next_node. }else { write curr_node code and curr_sym} if next_node is equal to MAX_CODE call trie_rest, and update prev_sym to current_sym} if curr_node isn't the root trie node { write prev_node code and prev_sym.} write STOP_CODE, 0, use flush_pairs, and write_pairs, and close the files

decode.c (main):

- check if infile is accessible and use open() for infile and outfile, call read_header, call wt_create and initialize with an empty word, length of 0, at index EMPTY_CODE. call read_pair and for each pair { append result of read code to table at index next_code, call write_word, increment next_code and check if it's MAX_CODE. if so, call wt_reset and set next_code to START_CODE. } call flush_words and close files.

trie.c:

node_create:

- allocate space for a node, and set its code from the parameter. set children pointers to null

node_delete:

- free space for the node

trie_create:

- creates and allocates space for root trie node with EMPTY_CODE, and returns node pointer if successful

trie_reset:

- walk the tree and recursively delete children nodes until only the root remains.

trie_delete:

- delete whole tree from the root

trie_step:

- returns pointer to child representing sym

word.c:

word_create:

- allocates space for word based off len given

word_append_sym:

- append sym to array in word and return its pointer

word_delete:

- free memory for word

wt_create:

- allocates space for an array of MAX_CODE words and calls word_create for the first EMPTY_CODE word.

wt_reset:

- sets all words after the first to null

wt_delete:

- frees memory for each word in word table

io.c:

read_bytes:

- loop of calls to read until to_read or until there aren't any left

write_bytes:

- loop of calls to write until to_write or if no bytes are written

read_header:

- reads in size of file header bytes from input into the header.

write_header:

-

read_sym:

-

write_pair:

-

flush_pairs:

-

read_pair:

-

write_word:

-

flush_words:

-