Wendi Tan
wtan37@ucsc.edu

Assignment 7

This assignment could compress and decompress files with lossless compression – Lempel-Ziv Compression. The compression algorithms encode data with repeated patterns using code-symbol pair. Code is an unsigned 16-bit integer, and symbol is an 8-bit ASCII character. We use trie and word to do encoding and decoding respectively. And the helper file io.c for reading and writing bytes to infile and outfile.

Trie:

```
1.trie_node_create(code){
        allocating memory for a TrieNode *n
        each n->children[i] = NULL
        n->code = code
        return n
}

2.trie_node_delete(*n){
        free(n)
}

3.trie_create(){
        TrieNode *root = trie_node_create(1)
        if(root){return root}
        return NULL
}

4.trie_delete(*n){
        if n is not NULL:
                for each ith child of n:
                        trie_delete(n->children[i])
                        n->children[i] = NULL
        trie_node_delete(n)
}

5.trie_reset(*root){
        for each ith child of n:
                trie_delete(root->children[i])
                root->children[i] = NULL
}
is not is not is not
```

trie_delete() is a recursive function, trie_reset() is non-recursive.
we need to use trie_delete inside trie_reset. trie_reset is just to delete all children under the ROOT, and trie_delete is the one who delete the children of called children and recursively, all the children are deleted.

```
6.trie_step(*n, sym){
        if n->children[sym] is not NULL:
                return n->children[sym]
        return NULL
}
```

Word:

```
1.word_create(*syms, len){
        allocating memory for Word *w
        w->len = len
        allocating memory for array w->syms
        for loop in len times:
                w->syms[i] = syms[i]
        if w is not NULL:
                return w
        return NULL
}

2.word_append_sym(Word *w, sym){
        Word *new_word = word_create(w->syms, w->len+1)
        new_word->syms = reallocate memory
        new_word->syms[w->len] = sym
        new_word->len = w->len+1
        return new_word
}

3.word_delete(*w){
        free(w->syms)
        w->syms = NULL
        free(w)
        w=NULL
}

4.WordTable *wt_create(){
        allocating memory for WordTable *wt
        wt[1] = word_create(NULL,0 )
}
```

```
5.wt_reset(*wt){
        for i in UINT16_MAX:
                free(wt[i]->syms)
                wt[i]->syms = NULL
                wt[i] = NULL
}

6.wt_delete(*wt){
        for i in UINT16_MAX:
                word_delete(wt[i])
                free wt[i]
                wt[i] is NULL
        free wt
        wt is NULL
}
```

I skipped pseudo codes for io.c, encode.c and decode.c is because all of them are followed up by codes either provided by TA or the instruction.