TokenizerT_:

- This structure has 3 variables:
  - char** TokenArray for dynamically storing all tokens, right and wrong
  - char** ResultArray for dynamically storing values of tokens, right and wrong
  - numTokens to keep track of number of tokens.

TKCreate(char*) :

- Opens file using char* argument.
- Allocates memory for tokenizer struct object
- Allocates memory for both arrays inside tokenizer structure
- Reads a line of 50 bytes.
- Checks if line is empty, if so gets new line
- Creates a char* pointer to the line
- Send this pointer to function TkGetNextToken(char*) and returns copy of token;
- Every time a token is returned, realloc space to both arrays inside tokenizer structure, and increment numTokens by one.
- After getting token, malloc memory with size of token + 1, and store into tokenArray;
- After getting token, increments pointer to line by: Length of token and no of white spaces found.
- Uses a while loop, send in new pointer and get token, until a null token has been found.
- Rereads a new line, and repeats process, until all lines are processed with.
- After reading every line, run a for loop with help of numTokens;
- Read each token from tokenArray;
  - Check what type of token it is, and store it into resultArray
- Close file, and return tokenizer structure pointer;

TKPrint(Tokenizer*):

- Iterate through a for loop, with help of numTokens inside tokenizer structure.
- For each iteration, write the token and result into a file(result). If result is an error, write it into different file. (error.msg)
- Close both files after loop and return;

TKDestroy(TokenizerT *):

- Iterate through a for loop, with help of numTokens inside tokenizer structure.
- For each iteration, free the token stored inside tokenArray[i];
- Check if resultArray[i] is an error code, and free that memory.
- After for loop, free memory allocated for both arrays inside structure
- Free memory for pointer of Tokenizer structure
- Set everything to NULL, and return.

5 Helper functions:

```
int is_empty(char*); //checks if string is empty
int checkFloat(char*);
int checkHex(char*);
int checkOctal(char*);
int checkDecimal(char*);
```

```
#include <stdio.h>          //standard
#include <stdlib.h>             //memory allocations
#include <string.h>             //USING this for checking if string is hex, strspn. Also
                                 used in other string manipulations.
#include <ctype.h>              //USING THIS FOR isspace() used in TKGetNextToken
```