CS2263 Assignment 4

Kohdy Nicholson

Design of my program in steps:

1. Gather html input from stdin, store in the input array.
2. Pre-process html input to remove all html comments from the input array.
3. Iterate over the html input, identifying tags which start with '<' follow by an alpha character.
4. Extract any identified tags into heap memory.
5. Check if the extracted tag is in the list of tags
    a. If so, increase the occurrence for that tag by 1, and free the tag memory as there is an identical copy already in the tag list.
    b. If not, add it to the tag list. Increase the occurrence and size of the tag list by 1.
6. Continue iterating over the list, repeating steps 4 and 5 until all html input is processed.
7. Loop through the list of tags, printing the strings and occurrences to the console. Free the tag afterwards.

About my functions:

- is_letter(): checks if the given character is a letter. This is because all html tags begin with a letter.
- is_num(): checks if the given character is a number. This is because some html tags have numbers in them such as h1-h6.
- extract_tag(): copies the tag name of an identified tag to the heap. Returns the pointer to the beginning of the string in the heap.
- check_tag_list(): Takes the list of tags and a tag as parameters. Checks if the tag exists in the list, if so, returns the index at which it exists. Otherwise, return -1.
- is_comment(): This checks if we have hit a comment. We use this before any html tag processing is done, so we can strip out the comments beforehand.
- remove_comment(): If is_comment() is equal to 1, we remove the contents inside the comment using the pointer which points to the beginning of the comment.

The use of the heap and strings made this assignment much easier than assignment 3. Although in assignment 3, I used the heap, I still had to make some changes to simplify the logic. These changes consisted of using a list of strings for the tag list. I also added a function for extracting a tag from the html input string. Therefore, we could use the strcmp function from the string.h module when checking to see if the tag already exists in the tag list. This made incrementing the occurrence and adding the tag to the list, much easier, and more readable (god bless).

htags.c source:

```c
/*
    htags.c

    Description:
    Program that counts the occurence of html tags from std input.

    Author:
    Kohdy Nicholson

    Date:
    2020-05-24
*/

#include "parser.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char** argv){
    // open html file specified on cmd line
    FILE* html = fopen(argv[1], "r");

    // String for storing the html input
    int n = 100000;
    char* contents = (char*)malloc(n);
    if(contents == (char*)NULL){
        fprintf(stderr,"Memory failure, terminating");
        return EXIT_FAILURE;
    }

    // list of tags
    char** tag_list = (char**)malloc(100);
    if(tag_list == (char**)NULL){
        fprintf(stderr,"Memory failure, terminating");
        return EXIT_FAILURE;
    }

    // Parallel integer table for the amount of occurences of each html tag in th
e list of tags
    int* occurences = (int*)malloc(100);
    if(occurences == (int*)NULL){
        fprintf(stderr,"Memory failure, terminating");
        return EXIT_FAILURE;
```

```
    }

    // variable for holding a pointer to the extracted tag in the heap
    char* tag;

    // Size of the contents string
    int size;

    // size of the list of tags
    int list_size;

    // index of the tag in the list
    int index;

    // variable for storing individual characters into the contents string
    char input;

    int i;

    /*

    *** STEP ONE: FILL THE contents ARRAY WITH HTML FILE CONTENTS ***

    */
    i = 0;
    while(fscanf(html, "%c", &input) != EOF){
        contents[i] = input;
        i++;
    }
    contents[i] = '\0'; //Null terminate the html contents

    /*

    *** STEP TWO: REMOVE CONTENT WITHIN COMMENT TAGS ***

    */
    size = strlen(contents);
    i = 0;
    while(i < size){ // First we need to remove comments from the html contents.
        if(contents[i] == '<'){
            if(is_comment(&contents[i])){
                remove_comment(&contents[i] + 4); // We add 4 to the base pointer
address to skip the comment start '<!--'
            }
        }
```

```
        i++;
    }

    /*

    *** STEP THREE: FIND FIRST OCCURENCE OF HTML TAGS ***

    */
    i = 0;
    list_size = 0;
    while(i < size){
        if(contents[i] == '<'){
            if(is_letter(contents[i+1]) == 1){

                /*

                *** STEP FOUR: FIND OTHER OCCURENCES OF HTML TAG. REMOVE TAGS ONC
E CONFIRMED.

                */

                tag = extract_tag(&contents[i+1]);
                index = check_tag_list(tag_list, tag, list_size);
                if(index == -1){
                    tag_list[list_size] = tag;
                    occurences[list_size] = 1;
                    list_size++;
                }else{
                    occurences[index]++;
                    free(tag); // We do not need this tag as there is an identica
l copy in the tag list.
                }
            }
        }
        i++;
    }

    /*

    *** STEP FIVE: PRINT DETAILS FOR HTML TAGS ***

    */
    i = 0;
    while(i < list_size){
        printf("%s", tag_list[i]);
```

```
        printf("\t\t%d\n", occurences[i]);

        free(tag_list[i]); // Free the memory, no longer needed.

        i++;
    }

    free(contents);
    free(tag_list);
    free(occurences);

    return EXIT_SUCCESS;
}
```

parser.c source:

```
/*
    parser.c

    Description:
    Functions for parsing html elements.

    Author:
    Kohdy Nicholson

    Date:
    2020-05-24
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/*
a cover function for malloc()
malloc and return memory for a string of stringsize characters
return (char*)NULL on failure
*/
char* mallocString(int stringsize){
    char* newString = (char*)malloc(stringsize);
    return newString;
}

/*
```

```
Function to check if the pointer contains a letter value
*/
int is_letter(char ch){
    int result = 0;
    if((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z')) result = 1;
    return result;
}

/*
Function to check if the pointer contains a number value
*/
int is_num(char ch){
    int result = 0;
    if(ch >= '0' && ch <= '9') result = 1;
    return result;
}

char* extract_tag(char* tag_start){
    char* tag_end = tag_start;
    char* tag;
    int stringSize = 1; // stringsize will be 1 even if emtpy, for the null chara
cter
    int i = 0;

    //This will loop through the input until the end of the tag name, capturing t
he stringSize as well
    while(is_letter(*tag_end) || is_num(*tag_end)){
        tag_end++;
        stringSize++;
    }

    tag = mallocString(stringSize);

    if(tag != (char*)NULL){
        // deep copy the tag into the newly allocated string on the heap
        while(tag_start != tag_end){
            tag[i] = *tag_start;
            tag_start++;
            i++;
        }

        tag[i] = '\0'; // null terminate the string
    }

    return tag;
```

```c
}

/*
Function which checks to see if the given tag is already in the list
returns the index if in the list, returns -1 if not;
*/
int check_tag_list(char** list, char* tag, int n){
    int i = 0;
    while(i < n){
        if(strcmp(list[i], tag) == 0){
            return i;
        }
        i++;
    }
    return -1;
}

/*
Function for checking to see if we have encountered a comment.
*/
int is_comment(char* start){
    int result = 0;

    if(*start == '<'){
        start++;
        if(*start == '!'){
            start++;
            if(*start == '-'){
                start++;
                if(*start == '-'){
                    result = 1;
                }
            }
        }
    }

    return result;
}

/*
Function for removing comment contents. Loop will only terminate when we reach th
e comment ending '-->'.
*/
void remove_comment(char* start){
    char* ending_dash = start + 1;
```

```
    char* ending_arrow = ending_dash + 1;
    while(*start != '-' || *ending_dash != '-' || *ending_arrow != '>'){

        *start = ' '; // This will begin removing everything inside the comment.

        start++;
        ending_dash++;
        ending_arrow++;
    }
}
```

Output on HelloWorld.html:

```
PS C:\Users\Kohdy\Documents\cs2263\W3Ass> Get-Content .\HelloWorld.html | .\
htags.exe
html            1
head            1
meta            1
title           1
body            1
p               1
PS C:\Users\Kohdy\Documents\cs2263\W3Ass>
```

Output on Sample.html:

```
PS C:\Users\Kohdy\Documents\cs2263\W3Ass> gcc .\htags.c .\parser.c -o htags
PS C:\Users\Kohdy\Documents\cs2263\W3Ass> Get-Content .\Sample.html | .\htag
s.exe
html            1
head            1
meta            1
title           1
body            1
strong          1
ol              1
li              2
blink           1
p               2
PS C:\Users\Kohdy\Documents\cs2263\W3Ass>
```

Output on David Bremner's CS3613 website home page, with the whole website also contained inside a comment (websiteception):

```
PS C:\Users\Kohdy\Documents\cs2263\W3Ass> gcc .\htags.c .\parser.c -o htags
PS C:\Users\Kohdy\Documents\cs2263\W3Ass> Get-Content .\cs3613.html | .\htag
s.exe
html            1
head            1
meta            1
title           1
link            4
body            1
div             13
span            6
a               39
ul              5
li              21
table           1
tr              6
td              12
strong          6
p               9
h2              2
br              4
i               4
h3              2
PS C:\Users\Kohdy\Documents\cs2263\W3Ass>
```

Testing remove_comment() function and program:

```
[knichol1@id415m13 W3Ass]$ make
available command:
        make help                   (this command)
        make htags                  (to build your C program)
        make remove_comments_test   (to build the test C program)
        make test                   (to run every test case)
        make compound_test          (to run htags compound tests)
        make remove_comment_test    (to run test cases against the function to remove
_comments)
[knichol1@id415m13 W3Ass]$ make test
gcc -Wall -c htags.c
gcc -Wall -c parser.c
gcc -Wall -o htags htags.o parser.o
./htags < Data/Sample.html > Sample_test.result
./TestPassed.sh Sample_test.result Data/Sample_test.expected

######   Passed   ###### Sample_test.result is equal to Data/Sample_test.expected

./htags < Data/HelloWorld.html > HelloWorld_test.result
./TestPassed.sh HelloWorld_test.result Data/HelloWorld_test.expected

######   Passed   ###### HelloWorld_test.result is equal to Data/HelloWorld_test.expe
cted

./htags < Data/hello.html > hello_test.result
./TestPassed.sh hello_test.result Data/hello_test.expected

######   Passed   ###### hello_test.result is equal to Data/hello_test.expected

gcc -Wall -c remove_comments_test.c
gcc -Wall -o remove_comments_test remove_comments_test.o parser.o
./remove_comments_test < Data/Sample.html > remove_comment_test1.result
./TestPassed.sh remove_comment_test1.result Data/remove_comment_test1.expected

######   Passed   ###### remove_comment_test1.result is equal to Data/remove_comment_
test1.expected

./remove_comments_test < Data/HelloWorld.html > remove_comment_test2.result
./TestPassed.sh remove_comment_test2.result Data/remove_comment_test2.expected

######   Passed   ###### remove_comment_test2.result is equal to Data/remove_comment_
test2.expected

[knichol1@id415m13 W3Ass]$
```

remove_comment_test1.result output: