**Daniyal Khan**

**3765942  
  
CS-2263**

**Assignment #3**

**Source Code:**

**main.c**

#*include* <stdio.h>

#*include* <stdlib.h>

#*include* <string.h>

#*include* "tags.h"

# *define* *N* 100000

# *define* *T* 100

void *readInput*(char \*fileName, char \*inputArray, int \*length);

void *countTags*(char \*arr, int length);

void *printTags*(char \*\*tags, int \*tagCounts, int tagNum);

int *main*(int argc, char \*\*argv) {

*if* (argc < 2) {

*printf*("Usage: %s <html\_file>\n", argv[0]);

}

char \*inputArr = *malloc*(*N*);

int inputArrLength = 0;

*readInput*(argv[1], inputArr, &inputArrLength);

*countTags*(inputArr, inputArrLength);

*putchar*('\n');

*free*(inputArr);

}

void *countTags*(char \*arr, int length) {

char \*\*tags = *malloc*(*T* \* sizeof(char\*));

int \*tagCounts = *malloc*(*T* \* sizeof(int));

*memset*(tagCounts, 0, *T* \* sizeof(int));

int tagNum = 0;

char \*end = arr + length;

*while* (arr < end) {

*if* (\*arr == '<' && \*(arr+1) != '!' && \*(arr+1) != '/') {

*if* (!*exists*(tags, arr, tagCounts, &tagNum)) {

\*(tags+tagNum) = arr;

tagCounts[tagNum]++;

tagNum++;

}

}

arr++;

}

*printTags*(tags, tagCounts, tagNum);

*free*(tags);

*free*(tagCounts);

}

void *printTags*(char \*\*tags, int \*tagCounts, int tagNum) {

char \*\*end = tags + tagNum;

*while* (tags < end) {

char \*tagPtr = \*tags;

*while* (\*tagPtr != '>' && \*tagPtr != ' ' && \*tagPtr != '/') {

*if* (\*tagPtr != '<') {

*printf*("%c", \*tagPtr);

}

tagPtr++;

}

*putchar*('\t');

*printf*("%i", \*tagCounts);

*putchar*('\n');

tags++;

tagCounts++;

}

}

void *readInput*(char \*fileName, char \*inputArr, int \*length) {

FILE \*fp = *fopen*(fileName, "r");

*if* (fp != *NULL*) {

char ch;

*while* ((ch = *fgetc*(fp)) != *EOF* && \*length < *N* - 1) {

\*(inputArr + \*length) = ch;

(\*length)++;

}

}

*fclose*(fp);

}

**tags.h**

#*ifndef* *TAGS\_H*

#*define* *TAGS\_H*

int *exists*(char \*\*tags, char \*arr, int \*tagCounts, int \*tagNum);

#*endif*

**tags.c**

#*include* "tags.h"

int *exists*(char \*\*tags, char \*arr, int \*tagCounts, int \*tagNum) {

char \*\*end = tags + (\*tagNum);

*while* (tags < end) {

char \*tagPtr = \*tags;

char \*arrPtr = arr;

*while* (\*tagPtr && \*arrPtr &&

\*tagPtr != '>' && \*tagPtr != ' ' && \*tagPtr != '/' &&

\*arrPtr != '>' && \*arrPtr != ' ' && \*arrPtr != '/' &&

\*tagPtr == \*arrPtr) {

tagPtr++;

arrPtr++;

}

*if* ((\*tagPtr == '>' || \*tagPtr == ' ' || \*tagPtr == '/' ) &&

(\*arrPtr == '>' || \*arrPtr == ' ' || \*arrPtr == '/' ))

{

(\*tagCounts)++;

*return* 1;

}

tags++;

tagCounts++;

}

*return* 0;

}

**Separately testing “exists” function:**

#*include* <stdio.h>

#*include* <stdlib.h>

#*include* <string.h>

#*include* "tags.h"

#*define* *T* 100

int *main*() {

char \*\*tags = *malloc*(*T* \* sizeof(char \*));

int \*tagCounts = *calloc*(*T*, sizeof(int));

int tagNum = 0;

*// Create a fake tag in memory*

tags[0] = "<div>";

tagCounts[0] = 1;

tagNum++;

*// Test tag that already exists*

char \*testTag1 = "<div>";

int result1 = *exists*(tags, testTag1, tagCounts, &tagNum);

*printf*("exists('<div>') = %d (should be 1)\n", result1);

*// Test tag that doesn't exist*

char \*testTag2 = "<span>";

int result2 = *exists*(tags, testTag2, tagCounts, &tagNum);

*printf*("exists('<span>') = %d (should be 0)\n", result2);

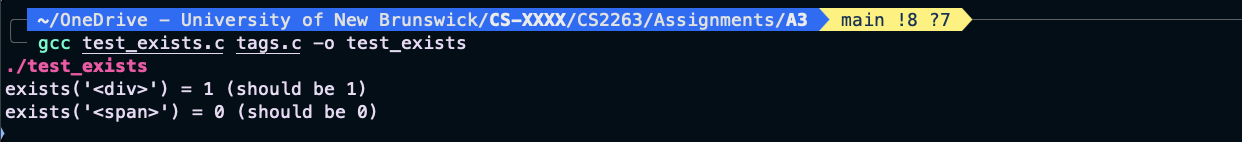
*free*(tags);

*free*(tagCounts);

*return* 0;

}

**Test output:**



**Design of the program:**

The program reads an HTML file into a dynamically allocated character array using readInput(). It then parses the input using countTags(), which identifies opening tags and stores pointers to each unique tag in a heap allocated array of strings (tags). Alongside an integer array (tagCounts) tracks how many times each tag appears. The function exists() checks whether a tag has already been seen and updates its count if so. Finally, printTags() prints each tag and its count by iterating through the arrays.

Output of HelloWorld.html:

A computer screen with blue text

AI-generated content may be incorrect.

Output of Sample.html:

A screen shot of a computer

AI-generated content may be incorrect.

Output of Index.html:  
  
A screenshot of a computer

AI-generated content may be incorrect.