CS2263 Assignment 4
Stephen Cole
3553803

htags.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#include"htags.h"

#define MAX_TYPE_TAGS 99
#define MAX_TAG_SIZE 20
#define MAX_NUM_TAGS 5000
#define MAX_STRING_LENGTH 255

void instantiateInputArray(char** inputArr, char* line, FILE* fp)
{
	int i = 0;
	while(fgets(line, MAX_STRING_LENGTH, fp))
	{
		inputArr[i] = (char*)malloc(MAX_STRING_LENGTH);
		strcpy(inputArr[i], line);
		i++;
	}
}

int getlines(FILE* fp, char* line)
{
	int numLines=0;
	while(fgets(line, MAX_STRING_LENGTH, fp))
		numLines++;

	rewind(fp);

	return numLines;
}

void scanWord(char* start, char* word)
{
  int counter = 0;
  if(*start == '<')
    start++;
  while(*start != '>' && *start != ' ' && *start != '/')
  {
    word[counter] = *start;
    counter++;
    start++;
  }
  word[counter] = '\0';
  return;
}
```

htags.h

```
#ifndef INTEGER_H
#define INTEGER_H

void instantiateInputArray(char** inputArr, char* line, FILE* fp);
int getlines(FILE* fp, char* line);
void scanWord(char* start, char* word);

#endif
```

findtags.c

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<stdbool.h>
#include"htags.h"

#define MAX_TYPE_TAGS 99
#define MAX_TAG_SIZE 20
#define MAX_NUM_TAGS 5000
#define MAX_STRING_LENGTH 255

/*
 * This whole program is a mess
 *
 */
int main(int argc, char* argv[])
{
        FILE *fp;
        char *line = (char*)malloc(MAX_STRING_LENGTH);

        fp = fopen(argv[1], "r");
        if(!fp)
                perror(argv[1]), exit(1);

        int numLines = getlines(fp, line);
```

```
char** inputArr = (char**)malloc(numLines*sizeof(char*));
instantiateInputArray(inputArr, line, fp);

char** tagTable = (char**)malloc(MAX_NUM_TAGS*sizeof(char*));
for(int i=0;i < MAX_NUM_TAGS;i++){
        tagTable[i] = (char*)malloc(MAX_TAG_SIZE);
}

int tagIndex = 0;
for(int i=1; i<numLines; i++)
{
        char* tagLocation;
        tagLocation = strstr(inputArr[i], "<");
        if(tagLocation != NULL)
        {
                if(*(tagLocation + 1) == '/' || *(tagLocation + 1) == '!')
                {
                        if(*(tagLocation + 2) == '-')
                                continue;
                        tagLocation++;
                        tagLocation++;
                }

                scanWord(tagLocation, tagTable[tagIndex]);
                tagIndex++;

                tagLocation = strstr((tagLocation + 1), "<");
                while(tagLocation != NULL)
                {
                        if(*(tagLocation + 1) == '/' || *(tagLocation + 1) == '!')
                        {
                                if(*(tagLocation + 2) == '-')
                                        break;

                                tagLocation++;
                                tagLocation++;
                        }

                        scanWord(tagLocation, tagTable[tagIndex]);
                        tagIndex++;
                        tagLocation = strstr((tagLocation + 1), "<");
                }
        }
}
```

```
	char** foundTags = (char**)malloc(MAX_TYPE_TAGS*sizeof(char*));
	for(int i=0;i < MAX_TYPE_TAGS;i++){
		foundTags[i] = (char*)malloc(MAX_TAG_SIZE);
	}

	int numFoundTags = 0;
	bool check = true;
	for(int i=0;i<tagIndex; i++)
	{
		for(int j=0;j<numFoundTags;j++)
		{
			if(strcmp(tagTable[i],foundTags[j]) == 0)
			{
				check = false;
				break;
			}
		}

		if(check)
		{
			strcpy(foundTags[numFoundTags],tagTable[i]);
			int tagCount = 1;

			for(int k=i+1; k<tagIndex; k++)
			{
				if(strcmp(foundTags[numFoundTags],tagTable[k]) == 0)
					tagCount++;
			}

			printf("%s\t%d\n",foundTags[numFoundTags], tagCount);
			numFoundTags++;
		}

		check = true;
	}


	fclose(fp);
	free(line);
	free(tagTable);
	free(inputArr);
	free(foundTags);

	return EXIT_SUCCESS;
}
```

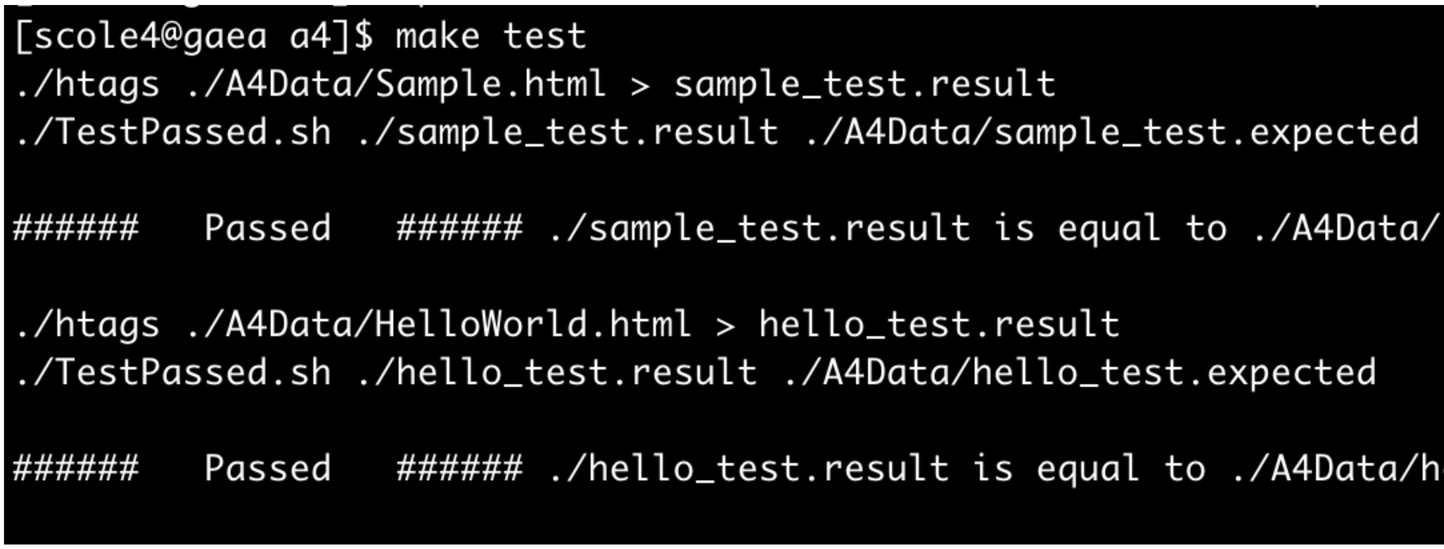
makefile

```
htags: findtags.c htags.c
	gcc -std=c99 -o htags -Wall findtags.c htags.c

test: sample_test hello_test

sample_test: htags
	./htags ./A4Data/Sample.html > sample_test.result
	./TestPassed.sh ./sample_test.result ./A4Data/sample_test.expected

hello_test: htags
	./htags ./A4Data/HelloWorld.html > hello_test.result
	./TestPassed.sh ./hello_test.result ./A4Data/hello_test.expected
```



```
[scole4@gaea a4]$ make test
./htags ./A4Data/Sample.html > sample_test.result
./TestPassed.sh ./sample_test.result ./A4Data/sample_test.expected

######   Passed   ###### ./sample_test.result is equal to ./A4Data/

./htags ./A4Data/HelloWorld.html > hello_test.result
./TestPassed.sh ./hello_test.result ./A4Data/hello_test.expected

######   Passed   ###### ./hello_test.result is equal to ./A4Data/h
```

outputs:

sample_test.result

```
html        2
head        2
meta        1
title       2
body        2
strong      2
ol          2
li          4
blink       2
p           4
```

hello_test.result

```
html                          2
head                          2
meta                          1
title                         2
body                          2
p                             2
```

The data structures in this program include, inputArr which is a string array containing each line from the html file passed in. tagTable is another string array containing every tag found in the inputArr. The last data structure is foundTags which contains the tags that have already been printed out.