CS 2263 - FR01A

Assignment 4

By Ngoc Phuong Anh Nguyen – 3712361

October 2021

Questions

Question 1: In a few sentences describe the design of your program. Focus on what each of the data structures holds and how each of the functions acts on them.

The program assigns the memory for 3 arrays: a char array named inputArray with size 100000 which stores data from html file, an integer array named countTag with 100 slots that store the number of each tag, and an array of pointers named nameTag with 100 slots that stores the pointer to the start of each unique tag.

Same with assignment 3, when the program is begun, it opens a file and copy data into inputArray. When the process is done, the program starts to run through each letter to find out a '<' (this sign is the beginning of tag). Next, it checks the next character if there is a '/' or '!'. If yes, the tags are ignored. Once the beginning of the line which has a valid tag is found, the program use clean() function to clear the line, and leave the tag. For instance, if a tag was "<p style="color:red">", the function would clean it, and transfer it to "<p>", since only "p" is the tag name.

Meanwhile, the valid tags are also count so that the quantities are also stored. When the counting process is done, the program use print() function to print out the tag name and remove the tag signs "<>", and also printout the quantity of each tag.

Question 2: Show the testing of one of the functions using a test program.

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

char* mallocString(int stringsize)
{
  return (char*)malloc(sizeof(char)* (stringsize));
}

char* duplicateString(char* s)
{
  char* sCopy = mallocString(strlen(s));
  if(sCopy != (char*)NULL)
  {
    strcpy(sCopy,s);
  }
  return sCopy;
}

int main(int argc, char* argv[])
{
  char* output;
  output = duplicateString("Hello World");
  if(output == (char*)NULL)
  {
    fprintf(stderr,"Memory failure, terminating");
    return EXIT_FAILURE;
  }
  printf("%s\n", output);
  free(output);
  return EXIT_SUCCESS;
}
```

Figure 1: Source Code of print() function in a test program.

```
[anguyen5@gc112m30 Assignment 4]$ gcc -c testing.c
[anguyen5@gc112m30 Assignment 4]$
[anguyen5@gc112m30 Assignment 4]$ gcc -o run testing.c
[anguyen5@gc112m30 Assignment 4]$ ./run
Hello World
[anguyen5@gc112m30 Assignment 4]$
```

Figure 2: Output of the test program.

Question 3: Show the output from running your program on the included HelloWorld.html file.

Question 4: Show the output from running your program on the included Sample.html file.

htags.c | Sample.html | form-al.html | testing.c | stringTest.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

int clear(char* currentTag)
{
  int i = 0;
  while(*(currentTag + i) !='>' && *(currentTag + i) !=' ' && *(currentTag + i) !='/' && *(currentTag + i) !='\0')
  {
    i++;
  }

  if(*(currentTag + i) != '>')
  {
    *(currentTag + i) = '>';
    int countOpen = 1;
    while(countOpen)
    {
      i++;
      if(*(currentTag + i) == '\0')
      {
        break;
      }
      if(*(currentTag + i) == '<')
      {
        countOpen++;
      }
      else if(*(currentTag + i) == '>')
      {
        countOpen--;
      }
      *(currentTag + i) = ' ';
    }
  }
  return 0;
}

bool compare(char* string1, char* string2)
{
  int i = 0;
  while(*(string1 + i) != '>' && *(string2 + i) != '>')
```

Figure 3: Source Code of htags.c

```
43    {
44      if(*(string1 + i) != *(string2 + i))
45      {
46        return false;
47      }
48      i++;
49    }
50    return *(string1 + i) == *(string2 + i);
51  }
52
53  int getIndex(char *currentTag, char **tagNameArray, int tagArraySize)
54  {
55    int i = 0;
56    while(i < tagArraySize)
57    {
58      if(compare(currentTag, *(tagNameArray + i)))
59      {
60        return i;
61      }
62      i++;
63    }
64    return tagArraySize;
65  }
66
67  void print(char *string)
68  {
69    int i = 1;
70    while(*(string + i) != '>')
71    {
72      printf("%c", *(string + i));
73      i++;
74    }
75    printf("\t");
76  }
77
78  char* mallocString(int stringsize)
79  {
80    return (char*)malloc(sizeof(char)* (stringsize));
81  }
82
83  char* duplicateString(char* s)
84  {
```

Figure 4: Source Code of htags.c

```
83  char* duplicateString(char* s)
84  {
85    char* sCopy = mallocString(strlen(s));
86    if(sCopy != (char*)NULL)
87    {
88      strcpy(sCopy,s);
89    }
90    return sCopy;
91  }
92  void freeAll(char* input, char** nameTag, int* countTag)
93  {
94    int i;
95    free(input);
96    for(i = 0; i< 100; i++)
97    {
98      free(nameTag[i]);
99    }
100   free(nameTag);
101   free(countTag);
102 }
103 int main(int argc, char* argv[])
104 {
105   FILE *file;
106   char* inputArray = NULL;
107   char **nameTag = NULL;
108   int *countTag = NULL;
109
110   inputArray = (char *)malloc(sizeof(char) * 100000);
111   nameTag = (char **) malloc(sizeof(char *) * 100);
112   countTag = (int *) malloc(sizeof(int) * 100);
113
114   int pointer = 0;
115   int j = 0;
116
117   file = fopen(argv[1],"r");
118
119   if(file == (FILE*)NULL)
120   {
121     fprintf(stderr, "Unable to open file %s\n", argv[1]);
122     return EXIT_FAILURE;
123   }
124
```

Figure 5: Source Code of htags.c

```
124
125     char c = fgetc(file);
126
127     while(c != EOF)
128     {
129       *(inputArray + pointer) = c;
130       pointer++;
131       c = fgetc(file);
132     }
133     *(inputArray + pointer) = '\0';
134
135     int tagArraySize = 0;
136     for(pointer = 0; *(inputArray + pointer) != '\0'; pointer++)
137     {
138       if(*(inputArray + pointer) == '<' && *(inputArray + pointer + 1) == '!')
139       {
140         clear(inputArray + pointer);
141       }
142       else if(*(inputArray + pointer) == '<' && *(inputArray + pointer + 1) != '/' && *(inputArray + pointer + 1) != '\0')
143       {
144         clear(inputArray + pointer);
145         int index = getIndex(inputArray + pointer, nameTag, tagArraySize);
146         if(index == tagArraySize)
147         {
148           tagArraySize++;
149           *(nameTag + index) = duplicateString(inputArray + pointer);
150           *(countTag + index) = 1;
151         }
152         else
153         {
154           *(countTag + index) += 1;
155         }
156       }
157     }
158
159     pointer = 0;
160     while(pointer < tagArraySize)
161     {
162       print(*(nameTag + pointer));
163       printf("%d\n", *(countTag + pointer));
164       pointer++;
165     }
166
167     fclose(file);
168     freeAll(inputArray, nameTag, countTag);
169     return EXIT_SUCCESS;
170 }
```

Figure 6: Source Code of htags.c

```
[anguyen5@gc112m30 Assignment 4]$ gcc -c htags.c
[anguyen5@gc112m30 Assignment 4]$ gcc -o run htags.c
[anguyen5@gc112m30 Assignment 4]$ ./run HelloWorld.html
html    1
head    1
meta    1
title   1
body    1
p       1
[anguyen5@gc112m30 Assignment 4]$ ./run Sample.html
html    1
head    1
meta    1
title   1
body    1
strong  1
ol      1
li      2
blink   1
p       2
[anguyen5@gc112m30 Assignment 4]$
```

Figure 7: Output of Question 3 and 4

```
[anguyen5@gcl12m30 Assignment 4]$ ./run form.html
html    1
head    1
title   1
meta    1
body    1
form    1
input   2
br      1
[anguyen5@gcl12m30 Assignment 4]$ ./run index.html
html    1
head    1
meta    2
title   1
body    1
link    1
script  2
style   1
div     1
h1      1
small   1
p       3
a       28
h2      1
ul      1
li      26
span    26
hr      1
em      1
[anguyen5@gcl12m30 Assignment 4]$ ./run form-al.html
html    1
head    1
title   1
meta    1
body    1
form    1
input   5
br      4
select  1
option  5
textarea        1
```

Figure 8: Output of htags.c using form.html, form-al.html and index.html

Note: There is a problem when I run file with form-al.html. It did show the correct result, but it included this:

```
*** Error in `./run': corrupted size vs. prev_size: 0x0000000001274500 ***
======= Backtrace: =========
/lib64/libc.so.6(+0x7f474)[0x7fb4621d0474]
/lib64/libc.so.6(+0x8156b)[0x7fb4621d256b]
./run[0x400adf]
./run[0x400dfe]
/lib64/libc.so.6(__libc_start_main+0xf5)[0x7fb462173555]
./run[0x400739]
======= Memory map: ========
00400000-00402000 r-xp 00000000 00:2a 12890968964                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 4/run
00601000-00602000 r--p 00001000 00:2a 12890968964                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 4/run
00602000-00603000 rw-p 00002000 00:2a 12890968964                        /home1/ugrads/anguyen5/CS 2263/Assignments/Assignment 4/run
0125a000-01293000 rw-p 00000000 00:00 0                                  [heap]
7fb45c000000-7fb45c021000 rw-p 00000000 00:00 0
7fb45c021000-7fb460000000 ---p 00000000 00:00 0
7fb461f3b000-7fb461f50000 r-xp 00000000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7fb461f50000-7fb46214f000 ---p 00015000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7fb46214f000-7fb462150000 r--p 00014000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7fb462150000-7fb462151000 rw-p 00015000 fd:00 9964043                    /usr/lib64/libgcc_s-4.8.5-20150702.so.1
7fb462151000-7fb462315000 r-xp 00000000 fd:00 237296                     /usr/lib64/libc-2.17.so
7fb462315000-7fb462514000 ---p 001c4000 fd:00 237296                     /usr/lib64/libc-2.17.so
7fb462514000-7fb462518000 r--p 001c3000 fd:00 237296                     /usr/lib64/libc-2.17.so
7fb462518000-7fb46251a000 rw-p 001c7000 fd:00 237296                     /usr/lib64/libc-2.17.so
7fb46251a000-7fb46251f000 rw-p 00000000 00:00 0
7fb46251f000-7fb462541000 r-xp 00000000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7fb462719000-7fb46271c000 rw-p 00000000 00:00 0
7fb46273c000-7fb46273d000 rw-p 00000000 00:00 0
7fb46273e000-7fb462740000 rw-p 00000000 00:00 0
7fb462740000-7fb462741000 r--p 00021000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7fb462741000-7fb462742000 rw-p 00022000 fd:00 3016479                    /usr/lib64/ld-2.17.so
7fb462742000-7fb462743000 rw-p 00000000 00:00 0
7ffe53791000-7ffe537b2000 rw-p 00000000 00:00 0                          [stack]
7ffe537cb000-7ffe537cd000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0                  [vsyscall]
Abort
[anguyen5@gcl12m30 Assignment 4]$
```