

Assignment 4

1)

The htags3 program is designed to extract unique HTML tags from an HTML file provided as a command-line parameter. It utilizes dynamic memory allocation on the heap to store the detected tags in a 2D array. The program initially allocates a small chunk of memory for the array and dynamically reallocates memory as needed to accommodate additional tags. Each time memory is allocated or reallocated, the program prints the amount of the new allocation in bytes and the total memory allocated so far. The algorithm for parsing HTML tags remains similar to previous versions, scanning the file line by line and identifying tags enclosed within < and > characters while skipping over comment tags. Detected tags are stored in the dynamically allocated array. Finally, before termination, the program deallocates (frees) all the heap memory used.

2)

#source code

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

#define INITIAL_SIZE 100
#define INCREMENT_SIZE 100

char **tags = NULL;
int tagsCount = 0;
int totalAllocatedMemory = 0;

int isComment(const char *line) {
    return (line[0] == '<' && line[1] == '!' && line[2] == '-' &&
line[3] == '-');
}

void extractTags(const char *line) {
    const char *start = line;
    while ((start = strchr(start, '<')) != NULL) {
        const char *end = strchr(start, '>');
        if (end == NULL)
            break;
        if (!isComment(start)) {
            int length = end - start + 1;
            char *tag = (char *)malloc(length + 1);
            if (tag != NULL) {
                strncpy(tag, start, length);
                tag[length] = '\0';
                tagsCount++;
                if (tagsCount == 1) {
                    tags = (char **)malloc(INITIAL_SIZE * sizeof(char
*));
                }
            }
        }
    }
}
```

```

        totalAllocatedMemory += INITIAL_SIZE * sizeof(char
*);
        } else if (tagsCount % INITIAL_SIZE == 0) {
            tags = (char **)realloc(tags, (tagsCount +
INCREMENT_SIZE) * sizeof(char *));
            totalAllocatedMemory += INCREMENT_SIZE *
sizeof(char *);
        }
        tags[tagsCount - 1] = tag;
        printf("Allocated %d bytes. Total allocated memory: %d
bytes.\n", length + 1, totalAllocatedMemory);
        } else {
            printf("Memory allocation failed for tag: %s\n",
start);
        }
    }
    start = end + 1;
}

void printUniqueTags() {
    int i = 0;
    do {
        int j = 0;
        do {
            if (strcmp(tags[i], tags[j]) == 0)
                break;
            j++;
        } while (j < i);
        if (i == j)
            printf("%s\n", tags[i]);
        i++;
    } while (i < tagsCount);
}

void freeMemory() {
    for (int i = 0; i < tagsCount; i++) {
        free(tags[i]);
    }
    free(tags);
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL) {
        printf("Error opening file %s\n", argv[1]);
        return 1;
    }
}

```

```

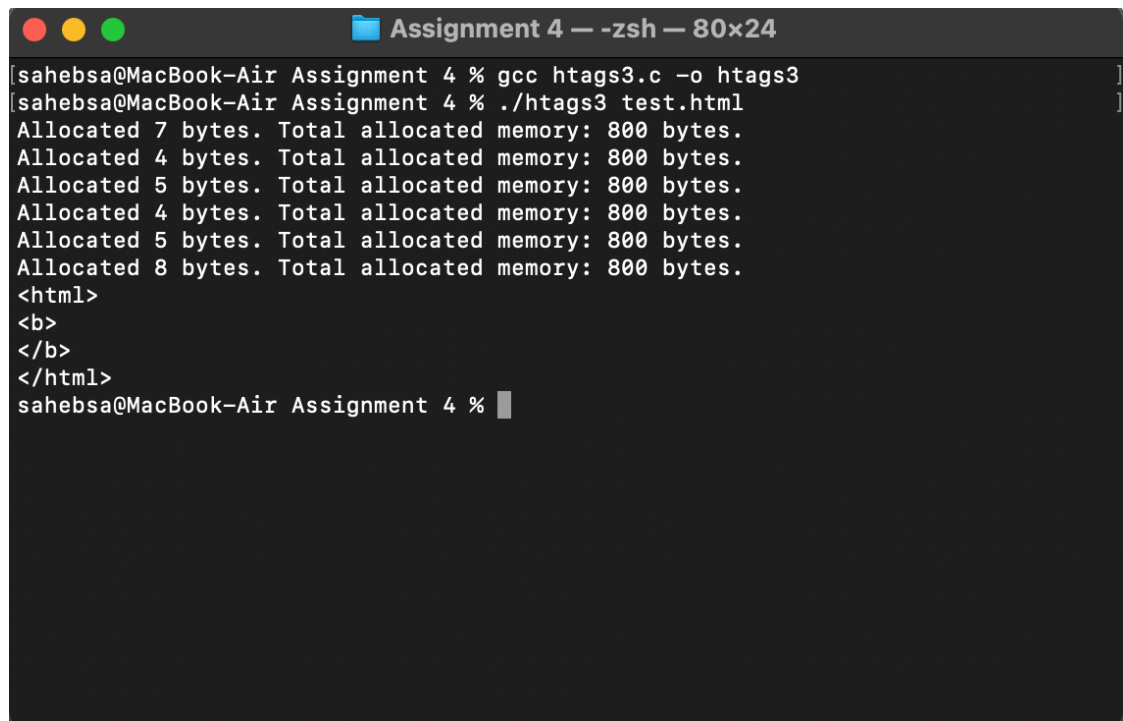
    char line[BUFSIZ];
    while (fgets(line, sizeof(line), file)) {
        extractTags(line);
    }
    fclose(file);

    printUniqueTags();
    freeMemory();

    return 0;
}

```

3)



A terminal window titled "Assignment 4 — -zsh — 80x24" shows the following commands and output:

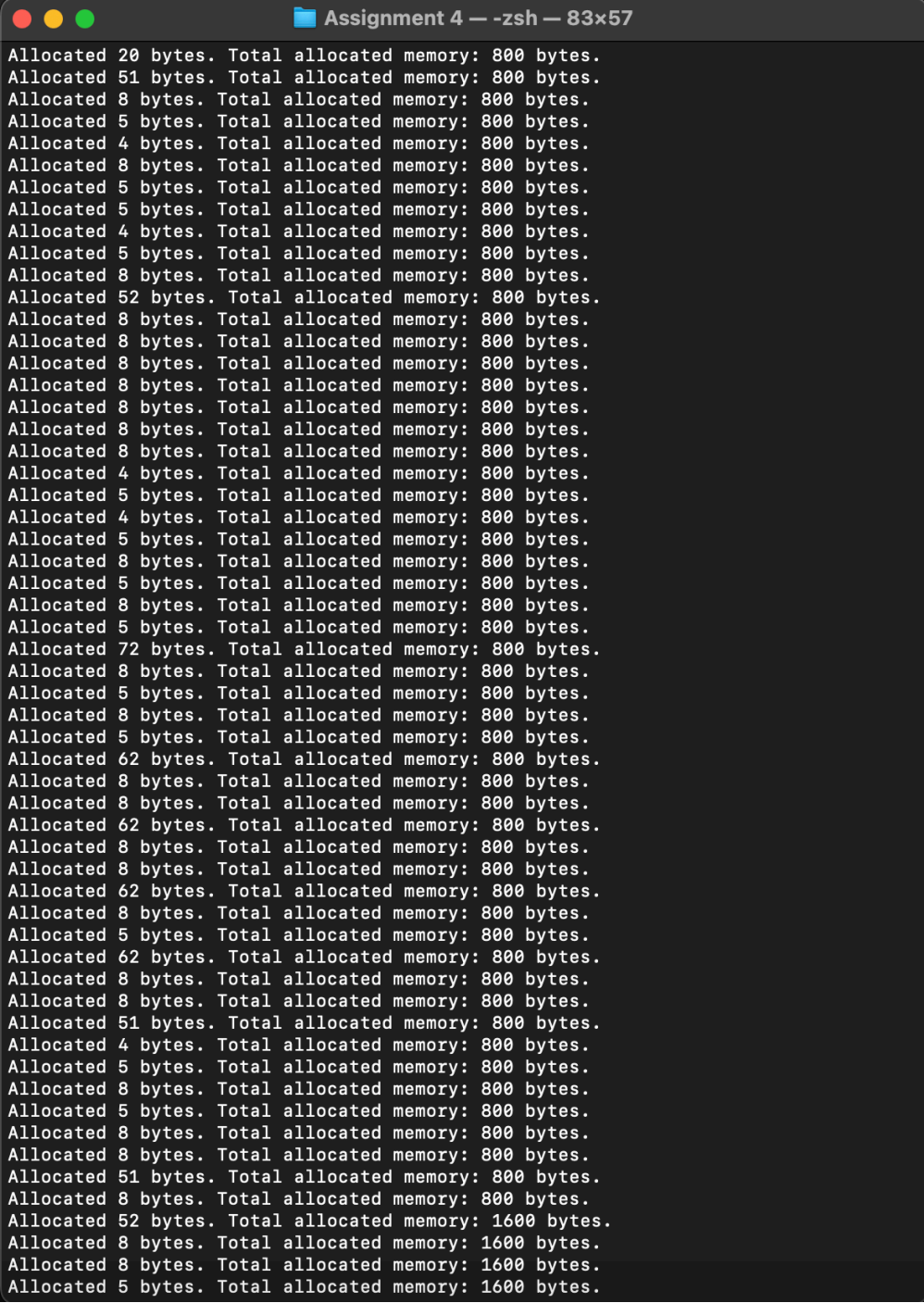
```

[sahebsa@MacBook-Air Assignment 4 % gcc htags3.c -o htags3
[sahebsa@MacBook-Air Assignment 4 % ./htags3 test.html
Allocated 7 bytes. Total allocated memory: 800 bytes.
Allocated 4 bytes. Total allocated memory: 800 bytes.
Allocated 5 bytes. Total allocated memory: 800 bytes.
Allocated 4 bytes. Total allocated memory: 800 bytes.
Allocated 5 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
<html>
<b>
</b>
</html>
sahebsa@MacBook-Air Assignment 4 %

```

4

[illegible]



```
Assignment 4 — -zsh — 83x57
Allocated 51 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
Allocated 52 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 51 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 52 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 51 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 52 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 20 bytes. Total allocated memory: 1600 bytes.
Allocated 62 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 5 bytes. Total allocated memory: 1600 bytes.
Allocated 7 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
Allocated 8 bytes. Total allocated memory: 1600 bytes.
<html>
<head>
<meta http-equiv=Content-Type content="text/html; charset=windows-1252">
<meta name=Generator content="Microsoft Word 15 (filtered)">
<style>
</style>
</head>
<body lang=EN-CA link=blue vlink="#954F72" style='word-wrap:break-word'>
<div class=WordSection1>
<p class=MsoNormal>
<b>
</span>
```



```

</span>
</b>
</p>
<p class=MsoNormal style='text-align:justify'>
<u>
<span lang=EN-US style='font-family:"Times New Roman",serif'>
</u>
<a href="https://www.educba.com/types-of-tags-in-html/">
</a>
<span style='font-family:"Times New Roman",serif'>
<i>
</i>
<span lang=EN-US style='font-family:"Courier New"'>
<p class=MsoBodyTextIndent style='margin-top:6.0pt;line-height:12.0pt'>
<br>
</div>
</body>
</html>
sahebsa@MacBook-Air Assignment 4 % █

```

5)

```

Assignment 4 — -zsh — 83x57
[sahebsa@MacBook-Air Assignment 4 % gcc htags3.c -o htags3 ]
[sahebsa@MacBook-Air Assignment 4 % ./htags3 my.html ]
Allocated 7 bytes. Total allocated memory: 800 bytes.
Allocated 7 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
Allocated 9 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
Allocated 7 bytes. Total allocated memory: 800 bytes.
Allocated 5 bytes. Total allocated memory: 800 bytes.
Allocated 6 bytes. Total allocated memory: 800 bytes.
Allocated 4 bytes. Total allocated memory: 800 bytes.
Allocated 5 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
Allocated 8 bytes. Total allocated memory: 800 bytes.
<html>
<head>
<title>
</title>
</head>
<body>
<h1>
</h1>
<p>
</p>
</body>
</html>
sahebsa@MacBook-Air Assignment 4 % █

```