CS 240: Programming in C
Fall 2020
Homework 10

Prof. Turkstra

Due November 18, 2020 9:00 PM

1 Goals

The purpose of this homework is to give you experience in using binary trees. No I/O this time. We'll just focus on the data structures.

Recursion

Part of the purpose of this homework is to gain experience with recursive functions. `insert_node()`, `search_node()`, `delete_tree()` and `find_message()` must be implemented recursively to receive any credit!

2 The Big Idea

In this homework, you will use the recursive concept to work on a binary tree and decrypt a secret message from the tree. You will first implement functions like `create_node()`, `insert_node()`, `search_node()`, `delete_tree()`. After completing them, you will implement the decrypt message functions to find the secret message that hides in the binary tree.

Each tree node contains a different int value (`hash_value`) and a char (`hash_char`). Combining all the chars from each node with a specific traversal order can form the final secret message. You will ask to find the secret message by giving a specifal traversal order.

We will utilize the following structures:

```
typedef struct tree_node {
  int hash_value;
  char hash_char;
  struct tree_node *left_child_ptr;
  struct tree_node *right_child_ptr;
} tree_node_t;
```

There are several ways of traverse a binary tree, including :

1. PREFIX (Current - Left - Right);
2. POSTFIX ((Left - Right - Current);
3. FORWARD (Left - Current - Right);
4. REVERSE (Right - Current - Left).

Here are the defined values for these traverse orders:

```
#define PREFIX  (1)
#define POSTFIX (2)
#define FORWARD (3)
#define REVERSE (4)
```

We also have an global variable that is used to store the message being traversed from the tree:

```
external char g_message[MAX_NODE_SIZE];
```

Examine hw10.h carefully. It contains a declaration for the tree type that you'll use. It also contains prototypes for the functions that you will write.

3 The Assignment

3.1 Functions you will write

You will write the following functions:

`tree_node_t *create_node(int, char);`

Dynamically allocate memory for a tree structure with both the hash value (first input argument) and hash char (second input argument) and initialize both `left_child_ptr` and `right_child_ptr` to NULL.

This function should return a pointer to the newly allocated node.

This function should assert that the first argument is a positive integer. Remember to also assert that the pointer returned by malloc() is not NULL.

`void insert_node(tree_node_t **, tree_node_t *);`

The first argument is a pointer to the pointer to the root tree element. The second argument is a pointer to the new tree element to be inserted. If the root tree is NULL, the root should be set to the newly inserted element. The tree should remain a binary tree after inserting the node. The key for insertion should be the `hash_value` field. (Assume that there are no duplicate hash values.)

You should assert that neither the first nor the second argument is NULL. You should assert that the second argument's `left_child_ptr` and `right_child_ptr` fields are NULL.

`tree_node_t *search_node(tree_node_t *, int);`

This function should find the node with the matching `hash_value` (second argument) in the binary tree given by the `root_node` (first argument) and return the pointer to the node found.

(Assume that there are no duplicate hash values.) It should return NULL if the given `root_node` is NULL or if the node with the given `hash_value` is not found.

This function should assert that the second argument is a positive integer.

```
void delete_tree(tree_node_t **);
```

Delete the entire tree and its associated data recursively. After the function returns, the root of the tree that was passed by pointer should be set to NULL.

The function should assert that the first argument is not NULL. However, keep in mind that it is okay for the first argument to point to a NULL pointer.

```
int find_message(tree_node_t *, int);
```

This function should traverse the tree from the root (first argument) using a traversal order (second argument) to find the secret message. When traversing the tree, store `hash_char` into `g_message` in order. And the function returns the total number of the tree nodes by the given root.

The function should assert that the second argument should be one of the traverse orders.

Hint: You can create your own helper functions and if the helper functions are implemented recursively, you will receive full credits as well.

3.2 Input Files

There are no input files for this assignment.

3.3 Header Files

We provide a header file, hw10.h, for you. It contains prototypes for each of the functions that you will write as well as #definitions for the constants. You should not alter this file. We will replace it with the original when grading.

3.4 Error Codes

There are no error codes for this assignment.

These Standard Rules Apply

- You may add any `#includes` you need to the top of your hw10.c file;
- You may not create any global variables other than those that are provided for you. Creation of additional global variables will impact your style grade;
- Do not look at anyone else's source code. Do not work with any other students.

Submission

To submit your program for grading, type:

```
$ make submit
```

In your hw10 directory. You can do this as often as you wish. We encourage you to submit your code as often as possible. Only your final submission will be graded.

4 Grading

The operation of your functions will be graded out of 100 points. The point breakdown will be determined by the test program.

The test program will be run many times when grading. It is your job to do the same. The lowest score will be your final grade.

This homework will also have a style grade based on 20 points, with 1 point deducted for each code standard violation found.

Your code must compile successfully using `-Wall -Werror -std=c99` to receive any credit. Code that does not compile will be assigned an automatic score of 0.

If your program crashes (e.g., segmentation fault) at any point during the testing process, your score will be assigned an automatic score of 0.