Texas Tech University Whitacre College of Engineering

CS 2413/5401 – Data Structures
Spring 2022
Lab Assignment 6

Acknowledge your collaborators or source of solutions, if any. **Online submission is required.**

While designing your programs or answering items, you are free to come up with your own assumptions based upon concepts and material learned in the course, if every potential specification is not given to you. Just be reasonable and document your assumptions. Such assumptions should not conflict with concepts and material learned in the course.

Your compliance with the "PROGRAMMING STYLE GUIDELINE" for CS 2413/5401 will affect your actual grade. All assignments will be checked for academic misconduct (cheating, plagiarism, collusion, falsifying academic records, misrepresenting facts, violations of published professional ethics/standards, and any act or attempted act designed to give unfair academic advantage to oneself or another student) defined by "OP 34.12: Grading Procedures, Including Academic Integrity" of TTU. Special software will be used to uncover such attempts.

A subset of answers submitted in this lab may be graded.

Objective: Practice using 2-3 search trees

Tasks:

1. You may work on this assignment by yourself, or you may work with one other student in this course as a team to complete this lab assignment.
    a. Teams larger than 2 people will incur a 25% penalty off the total lab points per extra person.
    b. It is expected that each team member contributes equitably and participates in coding and design ideas.
    c. If a team member is dissatisfied with the performance of the other team member, you are allowed to dissolve the team and continue individually.
        i. Whatever code each team member has contributed may be taken with that team member.
        ii. Try not to let such a decision wait for the day the assignment is due as no extensions will be given if a team is dissolved.
2. Write a C program, problem1.c, to read in a text file, called text.txt, and build a concordance file, called concordance.txt, for the file showing the number of unique words, the unique words, and the number of times each unique word occurs in the text file. A word only has alphabetic characters. Test your program with text.txt files of differing contents.
    a. Example
        i. Text File – text.txt (test empty, one word, and larger text files)
            1. A concordance of a text file is an alphabetical list of the unique words in the text file.
        ii. Concordance File – concordance.txt
            1. There are 13 distinct words in the text file:
            2. a 2
            3. alphabetical 1
            4. an 1
            5. concordance 1
            6. file 2
            7. in 1
            8. is 1
            9. list 1
            10. of 2

- **11.** text 2
- **12.** the 2
- **13.** unique 1
- **14.** words 1

- **b.** Use a 2-3 search tree to store the words and keep a count of the number of times each word occurs in the text file.
  - **i.** The 2-3 search tree operations, such as insert, search, count_nodes, count_keys, get_height, and others, should be written as separate functions.
  - **ii.** The root of the tree should be a pointer that points to the root node of the tree.
  - **iii.** Hints:
    - **1.** bundle the data, the word and its frequency, as a struct to make data movements easier
    - **2.** allow 3 keys in a tree node and 4 pointers for when splits need to occur
    - **3.** insert 2 keys into the 2-3 tree, then 3, then 4, …, printing the 2-3 tree each time to be sure it is building successfully
  - **iv.** Insert Algorithm (chapter 11 lecture slides 21-25)
    - **1.** If the tree is empty
      - **a.** make a new tree node
      - **b.** insert the (key,1) into the node
      - **c.** set the pointers on either side of the key to NULL
      - **d.** set the root to the new node
    - **2.** If the tree is not empty and the key is not in the tree
      - **a.** locate the leaf node at which the key should be inserted
      - **b.** insert the key into the leaf node
      - **c.** if the leaf node has 3 keys
        - **i.** split the leaf into 2 nodes n1 (low key) and n2 (high key)
        - **ii.** insert the middle key into the parent node with a pointer to n2 (n1 is already in the parent)
        - **iii.** if the parent has 3 keys
          - **1.** split the parent into 2 nodes n3 and n4
          - **2.** insert the middle key into its parent with a pointer to n4
          - **3.** continue splitting nodes in the insertion path up the tree until the insertion of a middle key is successful or the root is encountered and is split if necessary
- **c.** The main function should be a driver to call other functions to perform the required tasks to check each equation.
- **d.** The program should output to the screen the following "tree check" which should also be added to the end of program1.c in the comments for each test case. Check for one word in the tree and for one word not in the tree.

```
   i.  Tree Check:
  ii.    Tree Height:  3
 iii.    Tree Size  : 10 nodes
  iv.    concordance is found and has frequency 1
   v.
  vi.  Tree Check:
 vii.    Tree Height:  3
viii.    Tree Size  : 10 nodes
  ix.    water is not found
```

- **e.** No global variables should be used but define macro constants and typedef's may be used.

f. The C library files string.h and ctype.h may be used, but please use the safe string operations, such as *strncmp* in string.h.

g. Report: In the comments at the end of the program, give the following information:

i. Team Member Names

1. For each team member, detail the work on the program concerning specific work, test cases, and code and functions designed, implemented, and modified, such as

a. Name

i. void insert (node_t **tree, char w[]); - designed/implemented/modified

ii. created text.txt for test case 1, 2, 3, …

iii. …

2. A grade penalty of up to 25% of the total assignment points, which will be in addition to any other penalties, may be considered if inequitable contributions are made, not enough detail is present to be convincing, and/or all team members have the same list; i.e., all team members allegedly did the same thing (if both team members do the same thing – one team member is not needed).

ii. Test Cases and Status

1. example text.txt – passed/failed

2. text.txt with one word – passed/failed

3. empty text.txt – passed/failed

4. text.txt with > 500 total words – passed/failed

5. …

iii. 2-3 Search Tree Analysis

1. What would be the worst case for inserting n words into a 2-3 tree?

2. Big O of the best case for inserting n words into a 2-3 search tree

3. Big O of the storage requirements for 2-3 search tree with n keys

4. For the example text.txt file, the tree height is 3. What is the minimum possible height?

Learning Outcomes:

- Understand C program concepts, such as structs and pointers
- Understand how to implement dynamically allocated 2-3 search trees

Grading: 50 points

- Standard Deductions - 14 points
- Problem 1 – 36 points (problem1.c, , text.txt, concordance.txt) with the rest of the test cases as text1.txt, concordance1.txt, …)
  - 2-3 Search Tree – 12 points
  - Report – 12 points
  - Test Cases – 12 points

Due Date:

3/4/2022, 11:59pm (submitted on Blackboard by ONE team member)