Data Structures

Assignment 2

Deadline: 11:55 pm on Friday, October 18, 2024

Lead TAs: Talha Rehan & Sheraz Waseem

General Information

Welcome to the first programming assignment of this course. In this assignment, you are required to implement a doubly linked list, stack, and queue. Moreover, you will see how these data structures can be used to solve real-life problems. Marks distribution and submission instructions can be found towards the end of the document

Plagiarism Policy

1. Students must not share the actual program code with other students.
2. Students must be prepared to explain any program code they submit.
3. Students cannot copy code from the Internet or use any sort of AI.
4. All submissions are subject to automated plagiarism detection.
5. All submissions will be compared with a code file generated from AI tools such as ChatGPT.
6. Students are strongly advised that any act of plagiarism will be reported to the Disciplinary Committee.

Running Test cases

- Open the TestCases Folder on your Terminal (use cd command to navigate)
- Type in the compile command **“g++ –std=c++17 -w -o Test1 Test1.cpp”**
- Type in the running command **“./Test1”**
- Do this for all the Test Files by changing the number in the above commands.
- **DO NOT INCLUDE ANY COUT STATEMENTS IN YOUR CPP FILES (-2 Deduction).**

Task 1

General Trees

In this part, you will be applying what you learned in class to implement different functionalities of a tree efficiently. The basic layout of a general tree is given to you in the Tree.h file. The template node in Tree.h represents a node in the tree. The class Tree implements the general tree which contains a pointer to root and other function declarations

NOTE: A node in this tree is permitted to have any number of children

NO CREATION OF ADDITIONAL HELPER FUNCTION IS ALLOWED.

Write implementation for the following methods as described here.

Tree(shared_ptr> root) Simple default constructor.
shared_ptr> findKey(T key) Finds the node with the given key and returns a pointer to that node. NULL is returned if the key doesn't exist.
shared_ptr> findKeyHelper(shared_ptr> currNode, T key) Helper function to be used in findKey function
bool insertChild(shared_ptr> newNode, T key) ● Inserts the given node as the child of the given key. Returns true if the insertion is successful and false if the key doesn't exist. Insertion should also fail if another node with the same key as the new node already exists.

• If the node at the given key already has children, the new node should be added to the end of the children vector.
vector>> getAllChildren(T key) Returns all the children of the node with the given key. Should return an empty vector in case the node has no child or key doesn't exist.
int findHeight() Returns the height of the tree. NOTE: A tree with only root has height 0.
int findHeightHelper(shared_ptr> currNode) Helper function to be used in the findHeight function
void deleteTree() Deletes the entire tree.
bool deleteLeaf(T key) Delete node with given key if and only if it is a leaf node. Doesn't delete the root node, even if it is the only node in the tree. Returns true on success, false on failure.
shared_ptr> deleteLeafHelper(shared_ptr> currNode, T key) Helper function to delete the leaf node.

Task 2

DOMAIN NAME SYSTEM

The internet is in its nascent stages, rapidly evolving into a global network. With an increasing number of domains being registered, there's a pressing need to organize and manage these domains efficiently. You have been hired to design a hierarchical system for the Domain Name System (DNS), a critical component of the internet's infrastructure. Your task is to develop a DNS 1 hierarchy using the general tree structure implemented in Part 1 of your assignment. The DNS hierarchy will structure and categorize domains and subdomains in a way that reflects their relationships and dependencies.

Structure of the DNS Hierarchy:

1. Root Node:

At the top of the hierarchy is the ROOT node, representing the starting point of the DNS (already done for you).

2. TLD Nodes:

The next level contains nodes for Top Level Domains (TLDs) like .com, .org, .net, etc. These nodes are children of the ROOT node.

3. Domain Nodes:

Under each TLD node, there will be domain nodes. For example, under .com, there could be google, facebook, etc.

4. Subdomain Nodes:

Each domain node can have multiple subdomain nodes. For instance, under google, there could be maps, mail, etc.

5. Further Levels:

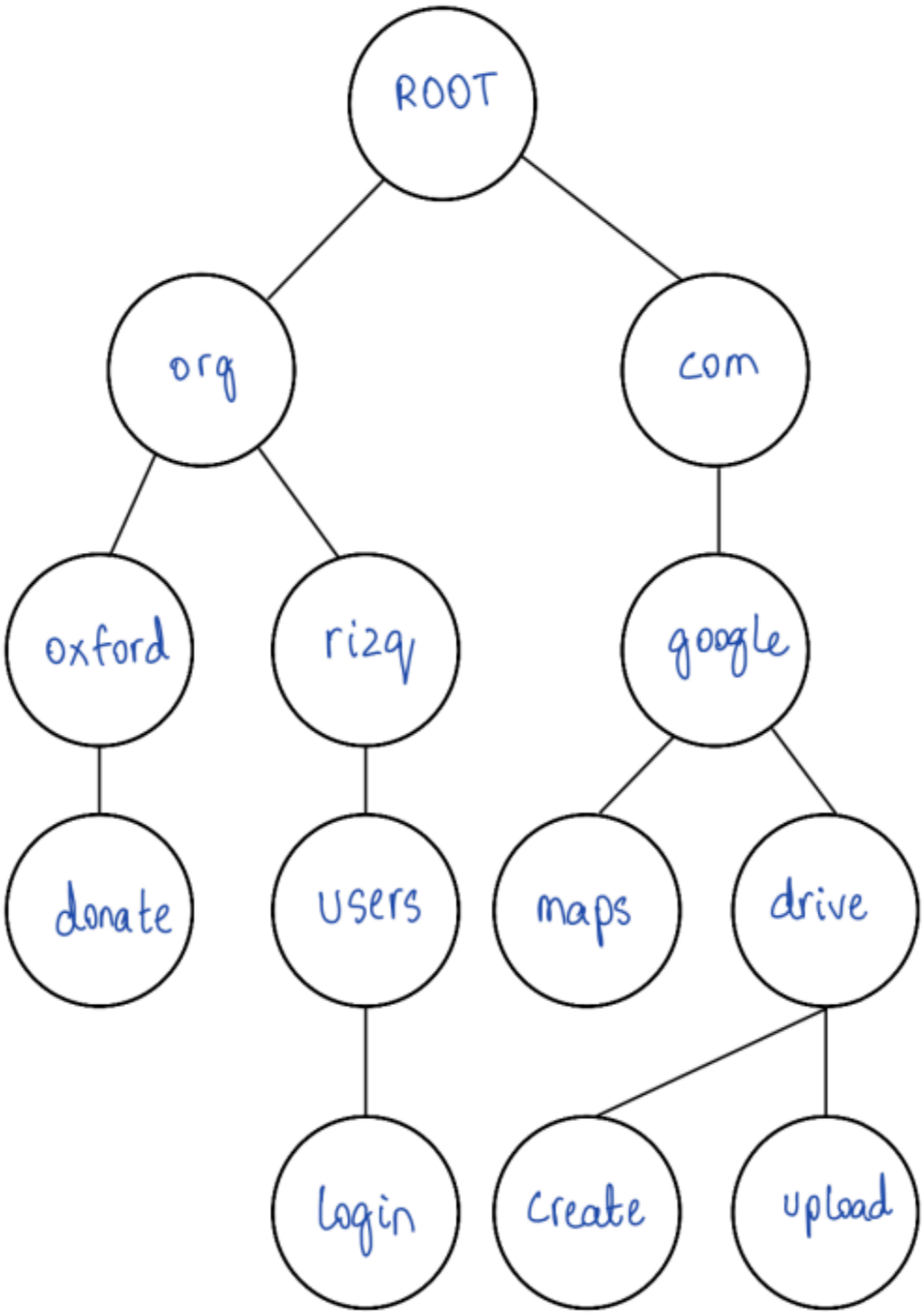
If applicable, subdomains can have their own child nodes, representing deeper levels of the URL structure, such as google.com/drive/create, where create is a child of drive.

Member functions:

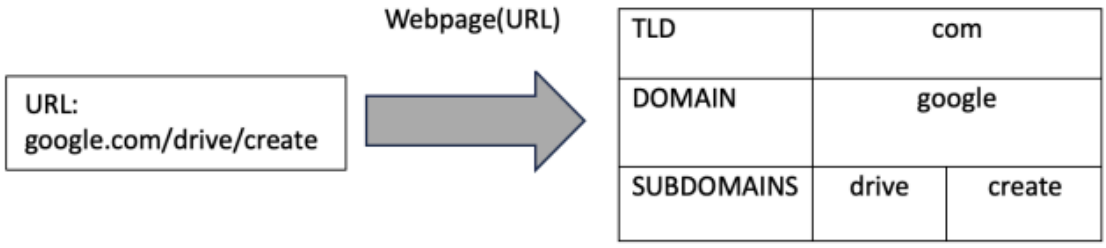
Write implementation for the following methods as described here

DNS() Simple default constructor which creates the ROOT node
void addWebpage(string url) Adds a new webpage to the DNS hierarchy. (Hint: Start from TLD, then domain name, and then sub domain)
int numRegisteredTLDs() Returns the count of nodes at the second level.
vector> getDomainPages(string domain) Retrieves all webpages under the specific domain.
void getDomainPagesHelper(shared_ptr> currDomain, string currentPath, shared_ptr> results) Helper function to be used in getDomainPages function.
vector> getAllWebpages(string TLD) Retrieves all webpages registered under the specific TLD.
shared_ptr> getDomainTree() Returns the pointer to the DNS tree
string findTLD(string domain) Finds the TLD for the given domain. Return an empty string if no such domain had been registered.
void generateOutput(vector> pages) Writes the content of the vector to the text file. Used for testing purposes

Here's a small demo of how the methods are supposed to work:



We have created a struct Webpage for you which parses the url into domain, subdomains, and TLD. An example:

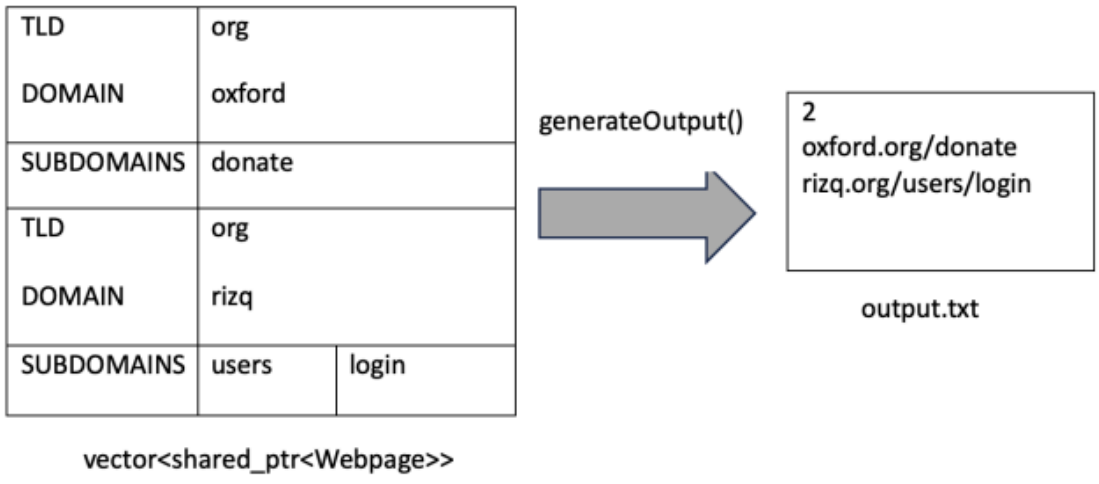


The tree on the previous page was created by the following function calls:

```
addWebpage("oxford.org/donate")
addWebpage("google.com/maps")
addWebpage("google.com/drive/create")
addWebpage("rizq.org/users/login")
addWebpage("google.com/drive/upload")
```

The function call below returns a vector:

1. `getAllWebpages("org")`



vector<shared_ptr<Webpage>>

2. `getDomainPages("google")`

TLD	com	
DOMAIN	google	
SUBDOMAINS	maps	
TLD	com	
DOMAIN	google	
SUBDOMAINS	drive	create
TLD	com	
DOMAIN	google	
SUBDOMAINS	drive	upload

vector<shared_ptr<Webpage>>

generateOutput()

```
3
google.com/maps
google.com/drive/create
google.com/drive/upload
```

output.txt

The rest of the methods are self explanatory:

1. `findTLD("rizq")` returns "org"
2. `numRegisteredTLDs()` returns 2

NOTE:

- **No** creation of additional helper functions is allowed.
- **Only** methods defined in Part 1 should be used for the implementation.
- **Visualizing** the tree structure on paper with sample domains is recommended to facilitate understanding and implementation.

Task 3

Balanced Binary Trees

You have been hired as an HR Executive at A.F. Ferguson Your responsibility is to manage and maintain the company's employee data and ensure that the right candidates are retained for potential promotions and internal recruitment. The company stores employee records using an **AVL tree** data structure, where each node represents an employee's record (e.g., name, contact information, skills, years with the company, current department). The tree is ordered by the job seeker's years of work experience. Refer to avl.hpp, which provides all the required definitions.

Write your implementation in avl.cpp using the provided boiler code.

For this part, the following simplified node class is provided to store the resume:

```
template <class T, class S>
class Node {
public:
      S fullName;
      S gender;
      T workExperience;
      shared_ptr<node> left;
      shared_ptr<node> right;
      Int height;

      node(S n, S g, T w) {
            this->fullName = n;
            this->gender = g;
            this->workExperience = w;
            left = NULL;
            right = NULL;
            height = 1;
      }
};
```

NOTE:

- You can assume that fullName is a string without spaces and does not contain any invalid characters.
- gender is represented as 'M' or 'F'.
- workExperience represents the work experience of the applicant in months.

Implement the AVL class to store the simplified resumes of all applicants. Member functions: Write implementation for the following methods as described here.

AVL(bool isAVL) Simple default constructor to initialize the isAVL flag. If the isAVL flag is set, insertions and deletions will follow AVL property. Otherwise, it will be a simple BST
void insertNode(shared_ptr> N) Inserts the given node into the tree such that the AVL (or BST) property holds.
void deleteNode(T k) ● Deletes the node with the given key such that the AVL (or BST) property holds. ● As convention, while deleting a node with 2 children, the new root should be selected from the left subtree.
shared_ptr> searchNode(T k) Returns the pointer to the node with the given key. NULL is returned if the key doesn't exist
shared_ptr> getRoot() Returns the pointer to the root node of the tree
int height(shared_ptr> p) Returns the height of the tree. NOTE: A tree with only root has height 1.

Hints:

- Start by implementing all operations for a simple BST.
- Make additional helper functions for balancing the tree and the left and right rotations.
- Don't forget to update the height of nodes after insertion/deletion.

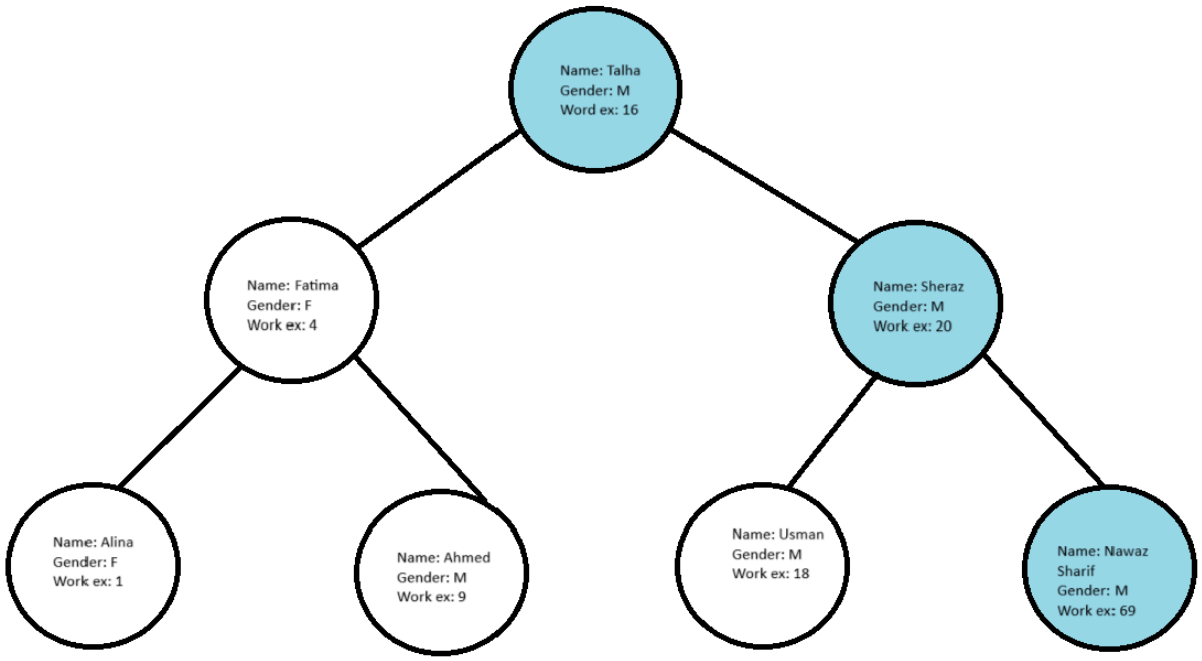
Task 4

Ethical Implication

The algorithm you use to shortlist applicants is as follows: Given a tree of resumes, all applicants that fall on the right-most path from the root to the leaf are shortlisted. The applicant at the highest level in the tree is given the highest priority. If some applicant cannot appear for an interview, you pick the left sibling of that applicant from the same level. Note that the number of shortlisted applicants equals the height of the rightmost leaf. You will be implementing the AVL class in ethics.cpp, defined in ethics.h.

The highlighted nodes in the following diagram are the shortlisted applicants. The priority of the shortlisted candidates is as follows (a lower integer means a higher priority):

1. Nawaz Sharif
2. Sheraz
3. Talha



Data analysts at Pakistan Tobacco apprise you of a surprising, yet unfortunate, insight. Based on the employee data collected from all Pakistan Tobacco offices, males tend to have a higher average work experience (when measured in number of months) than females. You realize that your shortlisting algorithm is biased against females! This is because job seekers with lesser work experience will generally occupy lower-left subtrees, thereby being less likely to be matched with job opportunities

PART 1

Your next task is to explore how this bias varies if you change your shortlisting algorithm. Your current approach has been stated above. You propose two alternatives:

Algorithm 1:

Shortlist applicants according to the in-order traversal of the tree

Algorithm 2:

Shortlist applicants according to the level-order traversal of the tree

You decide to write code to compare the two strategies. Implement the following functions:

int number_to_shortlist(shared_ptr> root) Returns the total number of applicants you can shortlist.
vector right_most(shared_ptr> root) Returns an array of keys on the right-most path from the root to the leaf
vector in_order(shared_ptr> root) Returns an array of keys traversed in-order from the root. Note that you can only shortlist a limited number of applicants; therefore, you need to return the same amount of keys as the number of applicants that can be shortlisted.
vector level_order(shared_ptr> root)

Returns a list of keys traversed in level-order from the root. Note that you can only shortlist a limited number of applicants; therefore, you need to return the same amount of keys as the number of applicants that can be shortlisted.

PART 2

You decide to raise your concerns about this bias to Pakistan Tobacco's CEO. However, he isn't convinced by your argument. To help solidify your argument, you decide to prove the existence of the bias (in the original algorithm).

Implement the following function to reveal the bias in the automated resume screening system

vector bias(shared_ptr> root)

- ● For every node in the tree, compute the ratio of Females to Males in the left subtree of that node.
- The function should return a vector which contains the ratios computed for all nodes, where the order of these ratios is the same as the level order traversal of the tree. (Simply put, the vector of ratios should be ordered the same way as the level-order traversal of the tree).

Additionally, include the answer to the following question as a comment (at the top of the function):

Describe what these ratios mean. Are they significant to prove that there is bias in the tree due to difference in average work experience?

Marks Distribution

Part	Marks
Task 1 – General Trees	**20**
Task 2 – Domain Name System	**30**
Task 3 – Balanced Search Tree	**30**
Task 4 – Ethical Implication	**20**

Submission guidelines

Zip the complete folder and use the following naming convention: PA1_rollnum.zip. For example, if your roll number is 26100026 then your zip folder's name should be: PA2_26100026.zip All submissions must be uploaded on LMS before the deadline. NO EXTENSIONS!

You are allowed 4 "free" late days during the semester (that can be applied to one or more assignments; the final assignment will be due tentatively on the final day of classes, i.e., before the dead week and cannot be turned in late. The last day to do any late submission is also the final day of classes, even if you have free late days remaining.