CSCI 2270: Data Structures

Recitation #7 (Section 101)

Office Hours

- Name: Himanshu Gupta
  Email: himanshu.gupta@colorado.edu

- **From next week onwards**
  - **Office Hours - 12pm to 2pm on Mondays in ECAE 128**
  - **Office Hours - 12:30pm to 2:30pm on Fridays in ECAE 128**

Office Hours Friday

Office Hours Friday

Office Hours

Do you need extra office hours this Friday for Assignment 5?

Logistics

Make Up Assignment

- You have an opportunity to redo one of the assignments in which you have scored less points and get 100% in that assignment.
- You can choose any assignment from Assignment 1 to Assignment 4.
- Make sure the assignment runs perfectly fine on your local system.
- In Interview Grading, I will ask basic questions to check that you understood what you have done and to ensure you have done that yourself.
  - When? - Office Hours. Send me an email to reserve a time slot for 10-15 mins during my office hours.
  - Needs to be done by the end of this week (February 27th and February 28th)

Logistics

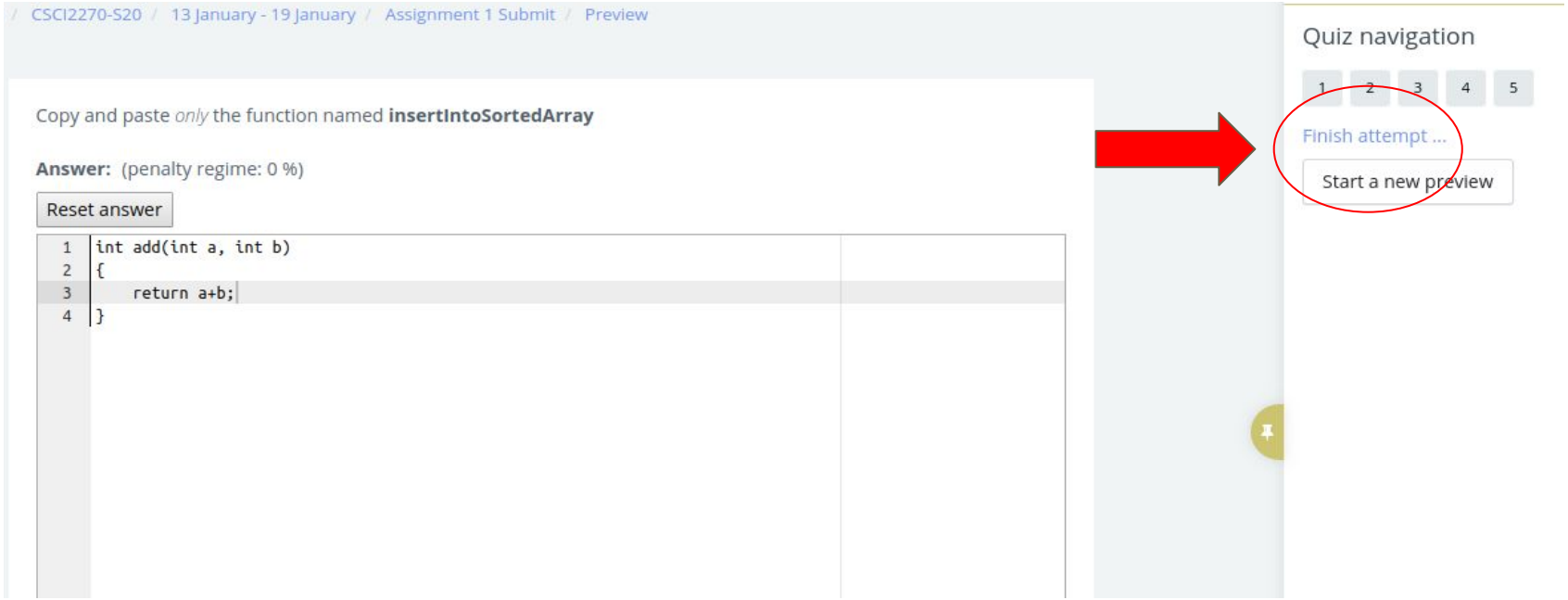
- Midterm 1 scores should be out on the weekend.
  - **Will be posted on Moodle.**
- Instructors have decided to not return the midterm papers.
- If you wish to see your paper, either shoot me an email or come to my office hours (This can be arranged next week).
- If you have any queries regarding grading, let me know and we will sort it out.

Logistics

- There will probably be another make up midterm towards the end of the semester.
  - It is not mandatory. You can take it if you wish to improve your grade.
  - I would advise anyone who gets <65 in Midterm 1 to take the make up midterm exam.

- **Assignment 5 is due on Sunday, March 1 2020, 11:59 PM.**
  GOOD LUCK!

Please click on “Finish Attempt” after you are done!



Any questions on Logistics?

Today's Agenda

- Review (20 ~ 30 mins)
  - Assignment 5
  - Common mistakes in Midterm 1
  - Trees
  - Recursion
  - Tree Traversal

- Exercise
  - **Silver Problem** - Implement "deleteTree" function which deletes all the nodes of the tree.
  - **Gold Problem** - Implement "sumNodes" function which returns the sum of all nodes present in the tree.

Assignment 5

How many people have started it or are done with it?

Assignment 5

- For those who haven’t, you are essentially just going to implement a stack using Linked List and a Queue (circular) using Arrays.
  - Do take care of the edge cases that arise when handling circular queues.
- Any specific problems people are running into?

Common Mistakes in Midterm 1

I found three common recurring mistakes.

Common Mistakes in Midterm 1

- Not covering edge cases, which leads to segmentation faults.

- For example:

  Not checking if the linked list is empty or not (head == NULL).

Common Mistakes in Midterm 1

- Memory Leak while traversing a linked list.

- For example:

```
Node * temp = head;
Node * temp = new Node;       //allocated memory from heap, but never deallocated it. Why do you even need to create a new node for traversing? You just need a pointer pointing to the head.
while(temp!= NULL){
        cout<<temp->data;
        temp = temp->next;
}
```

Common Mistakes in Midterm 1

- Not using Call by Reference or Call by Pointers to update the input array, its capacity and its size.

- void resize(int input_array, int capacity)
  //won't work because any changes made inside the function won't be reflected outside this function.

Any questions?

Trees

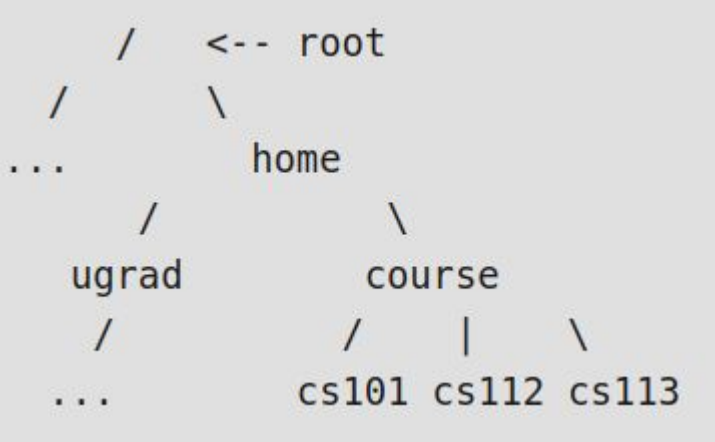
Trees

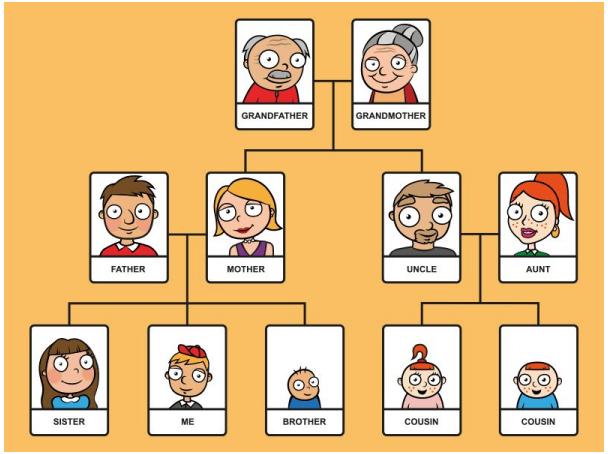
- Tree is a hierarchical or non-linear data structure.
  - Unlike Arrays, Linked Lists, Stacks and Queues which are linear.
- Why use trees?

Trees

- Tree is a hierarchical or non-linear data structure.
  - Unlike Arrays, Linked Lists, Stacks and Queues which are linear.

- Why use trees?
  - We might use trees when we want to store information that naturally forms a hierarchy. For example: the file system on a computer or your family tree or hierarchy of employees in an organisation.





Binary Trees

- Each node has at most two children. It has two pointers that point to the left and the right child of that node.

- The **root** of a tree is the node with no parents. Every tree has only one **root** node.

- A node with no children or NULL children is called a leaf node.

Binary Tree

What does a Tree node look like?

(We can similarly have a class declaration as well for a Node)

```
struct Node
{
  int data;
  Node *left;
  Node *right;
};
```

Recursion (Recursive functions)

Recursion

- A recursive function is a function that calls itself during its execution.
- This enables the function to repeat itself several times, outputting the result and the end of each iteration.
- **Having a termination criterion for your recursive function is of utmost importance.**
  - Or else what happens?

What's the output?

```
#include <iostream>
int fun(int n){
	if (n == 4)
		return n;
	else
		int num =  2*fun(n+1);
		return num;
}
int main(){
	cout<<fun(2);
	return 0;
}
```

A) 4
B) 8
C) 16
D) 2

Explanation

```
#include <iostream>
int fun(int n){
	if (n == 4)
		return n;
	else
		int num =
2*fun(n+1);
		return num;
}
int main(){
	cout<<fun(2);
	return 0;
}
```

```
Fun(2) = 2 * Fun(3) and Fun(3) = 2 * Fun(4) ....(i)
Fun(4) = 4 ......(ii)
From equation (i) and (ii),
Fun(2) = 2 * 2 * Fun(4)
Fun(2) = 2 * 2 * 4
Fun(2) = 16.
```

What's the output?

```
#include <iostream>
void fun(int n) {
	if (n == 0)
		return;
	cout<< n%2;
	fun(n/2);
}
int main(){
	fun(25);
	return 0;
}
```

(A) 11001
(B) 10011
(C) 11111
(D) 00000

What's the output?

```
#include <iostream>
void fun(int n) {
	if (n == 0)
		return;
	cout<< n%2;
	fun(n/2);
}
int main(){
	fun(25);
	return 0;
}
```

(A) 11001
(B) 10011
(C) 11111
(D) 00000

Any Questions?

Tree Traversal

- In order to do any operation on trees, we need a mechanism for visiting nodes of the tree data structure.

- The process of visiting all nodes of a tree is called tree traversal.

- Linear data structures like linked lists, stacks and queues the elements are visited in sequential order. However, in tree structures, there are many different ways.

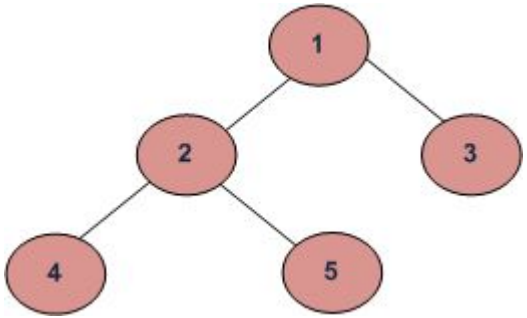
Tree Traversal

Mainly three different methods of traversing a tree.

- PreOrder Traversal
- InOrder Traversal
- PostOrder Traversal

PreOrder Traversal

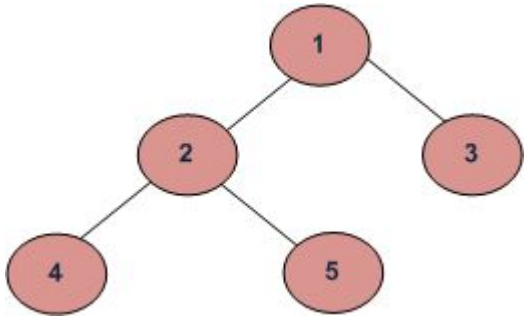
- Algorithm Preorder(tree)
  - Visit the current node.
  - Traverse the left subtree of current node, i.e., call Preorder(left-subtree)
  - Traverse the right subtree of current node, i.e., call Preorder(right-subtree)



Output for PreOrder Traversal?

PreOrder Traversal

```
void printPreorder(Node* node) {
        if (node == NULL)
                return;
         /* first print data of node */
        cout << node->data << " ";
        /* then recur on left subtree */
        printPreorder(node->left);
        /* now recur on right subtree */
        printPreorder(node->right);
}
```

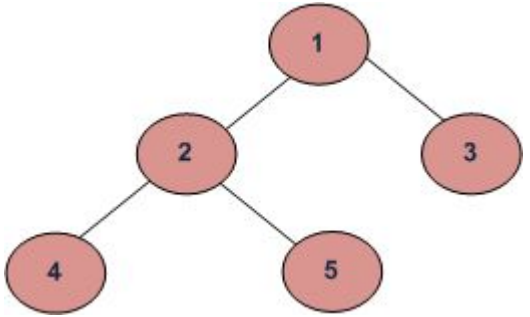


PreOrder - 1 2 4 5 3

InOrder Traversal

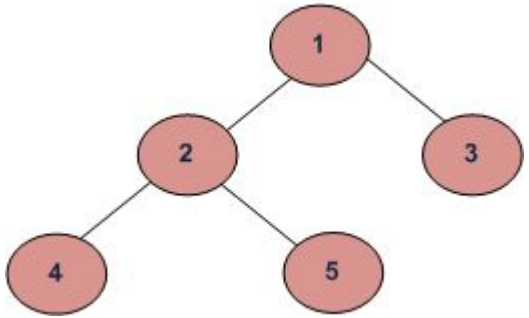
- Algorithm Inorder(tree)
  - Traverse the left subtree of current node, i.e., call Inorder(left-subtree)
  - Visit the current node.
  - Traverse the right subtree of current node, i.e., call Inorder(right-subtree)

Output for InOrder Traversal?



InOrder Traversal

```
void printInorder(Node* node) {
        if (node == NULL)
                return;
         /* first recur on the left child */
        printInorder(node->left);
        /* then print the data of the node */
        cout << node->data << " ";
        /* now recur on the right child */
        printInorder(node->right);
}
```

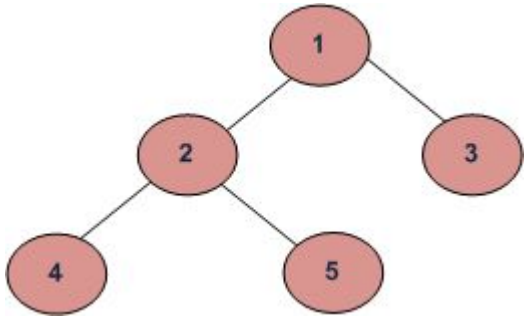


InOrder - 4 2 5 1 3

PostOrder Traversal

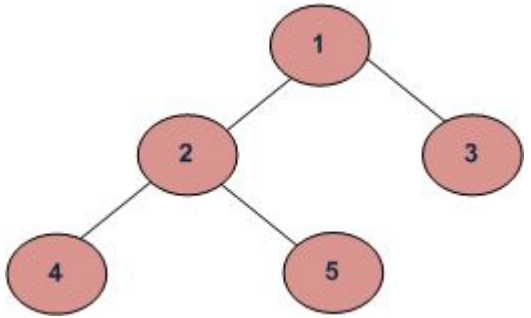
- Algorithm Postorder(tree)
  - Traverse the left subtree of current node, i.e., call Postorder(left-subtree)
  - Traverse the right subtree of current node, i.e., call Postorder(right-subtree)
  - Visit the current node.

Output for PostOrder Traversal?



PostOrder Traversal

```
void printPostorder(Node* node) {
        if (node == NULL)
                return;
         /* first recur on the left child */
        printPostorder(node->left);
        /* then recur on the right child */
        printPostorder(node->right);
        /* now print the data of the node */
        cout << node->data << " ";
        }
```



PostOrder - 4 5 2 3 1

Any questions on any traversal technique?

What's the output?

The inorder traversal of a binary tree is - d b e a f c g
The preorder traversal of a binary tree is - a b d e c f g.

What is the postorder traversal of the binary tree?

(A) d e b f g c a

(B) e d b g f c a

(C) e d b f g c a

(D) d e f g b c a

What's the output?

The inorder traversal of a binary tree is - d b e a f c g
The preorder traversal of a binary tree is - a b d e c f g.

What is the postorder traversal of the binary tree?

(A) d e b f g c a

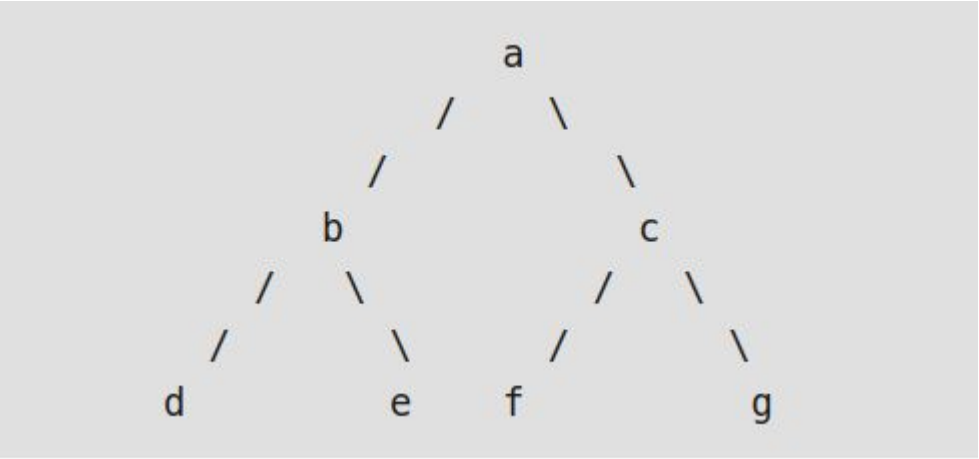
(B) e d b g f c a

(C) e d b f g c a

(D) d e f g b c a

What's the output?

The inorder traversal of a binary tree is - d b e a f c g
The preorder traversal of a binary tree is - a b d e c f g.



Few common errors

- Always check if your node is NULL or not.
  - How to do that?
  - Check if node == NULL (or node==nullptr)
    - If True, that means it is empty

- **REMEMBER: “root” word always used to refer to the first node of the tree (or the beginning if your tree).**

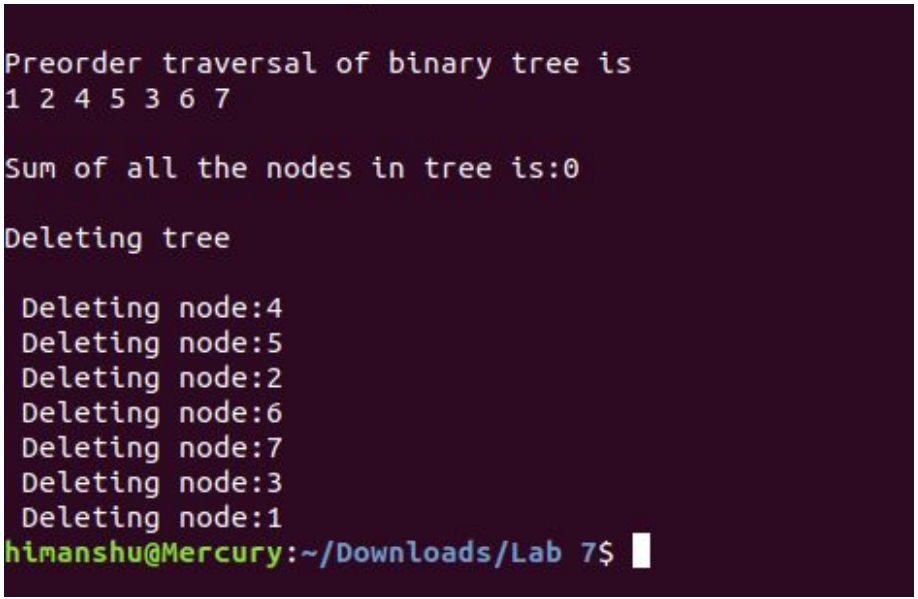
Exercise (Silver Problem)

- This problem is mandatory.
- Definition of the Tree class and Node struct is present in tree.hpp
  - I highly encourage you to read it.
- You will be implementing the Tree::deleteTree(Node *node) in tree.cpp
- Compile both driver.cpp and tree.cpp together for successful compilation.
  - **g++ -std=c++11 driver.cpp tree.cpp -o rec7**
  - **./rec7**

Exercise (Silver Problem)

- Which traversing technique should we be using and why?
- Think about the termination criterion of your recursive function.

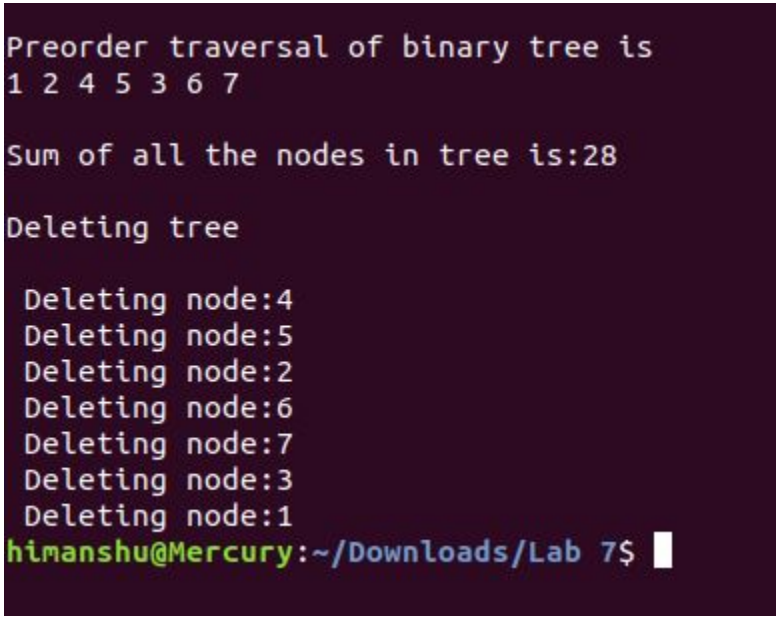
Expected Output



Exercise (Gold Problem)

- This problem is not mandatory but you are highly encouraged to solve it.
- You will be implementing the Tree::sumNodes(Node *node) function in tree.cpp
- Once again, compile both driver.cpp and tree.cpp together for successful compilation.
  - **g++ -std=c++11 driver.cpp tree.cpp -o rec7**
  - **./rec7**

Expected Output



Exercise (Gold Problem)

- **How can you solve this?**
  - Use a recursive function to traverse the tree. (Any traversing technique should work)
  - Take care of the edge case. What should happen if the node you are exploring is NULL? (Same as the termination criterion of your recursive function)

Exercise (Platinum Problem)

- This is not at all mandatory.
- Anyone who gets done with the recitation exercise early should try to attempt it.
- It is a common interview question asked by companies.
- Write down the pseudocode to generate postorder traversal of a binary tree if you are given the preorder and inorder traversal of that tree.

Exercise (Platinum Problem)

- **How can you solve this?**
  - Remember the first element in the preorder traversal is going to be the root of the tree, right? What's the second element in the preorder traversal?
  - The elements on the left of the root in inorder traversal are from the left subtree while elements on the right of the root in inorder traversal are from the right subtree