CSCI 2270 – Data Structures

Instructors: Zagrodzki, Ashraf

Midterm II

Part II – Coding question

Select only 1 of the following 2 questions. Extra credit will NOT be given for completing both questions. Starter code has been provided for each question in Canvas.

Question 1

Write C++ code that goes through a Binary Search Tree and deletes every leaf node with an odd key value. Your function should display the key values of each node of the tree, starting with the left child, then the right child, before displaying the root of each subtree. The nodes that are being deleted should display the key with a message "deleting" next to the key value. **Your code should prevent any potential memory leaks.**

Assume the following Node definition:

```
struct Node{
    int key;
    Node* left;
    Node* right;
    Node* parent;
};
```

You need to implement the following functions:

```
void delOddLeafHelper(<your arguments>)
{
}

void BST:: delOddLeaf(){
}
```

Note:

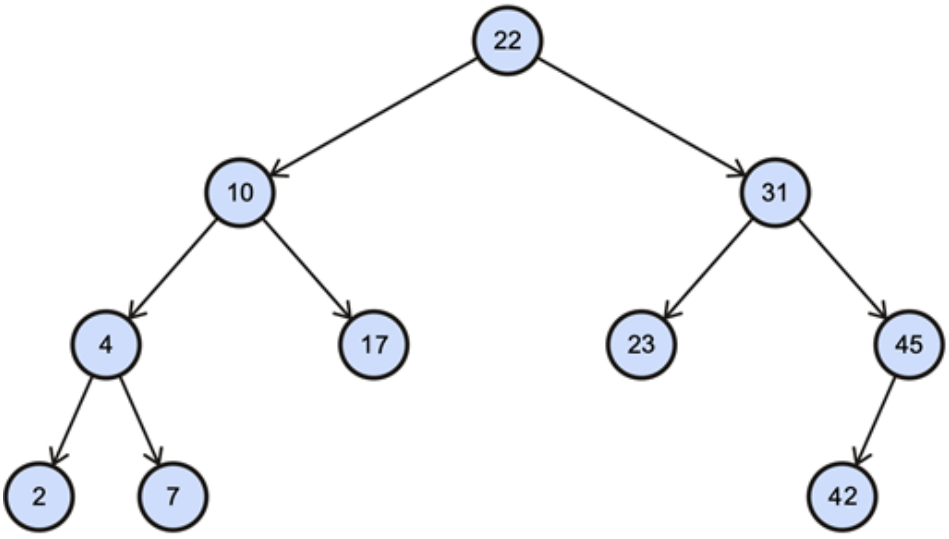
1. This function is part of a class Tree.
2. root is a ***private*** member of Tree class.
3. **You must use a helper function (deleteOddLeafHelper). You can pass arguments to this function as you desire. Do not change the hpp file. Add the helper in the cpp file.**

4. Your cout statement should be following. (Example tree images are below)
5. When you are ready to submit your solution, click the Canvas link named **Midterm II, Part II – Coding** under **Week 12** and upload all 3 of your solution files: BST.hpp, BST.cpp, and driver_bst.cpp.

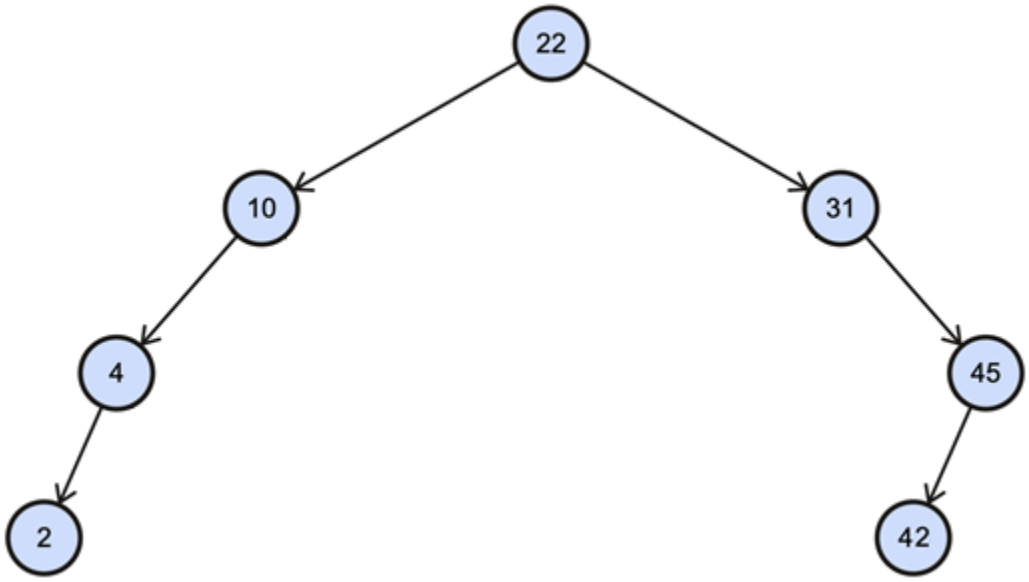
Terminal output:

2
7 (deleting)
4
17 (deleting)
10
23 (deleting)
42
45
31
22

Example: Before the deletion



After the deletion

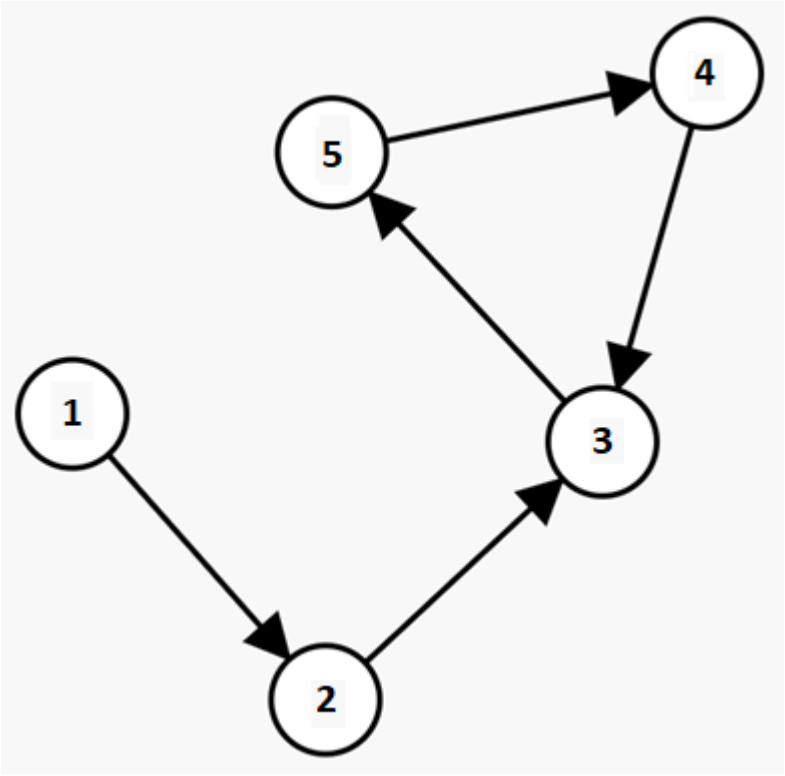


CSCI 2270 – Data Structures

Instructors: Zagrodzki, Ashraf

Question 2 (Remember, between Questions 1 and 2 you should only solve one)

Cycle detection: write a C++ function that for a directed graph, given a starting vertex, detects whether that vertex is a part of a cycle. A vertex is a part of a cycle if a path exists that will eventually return to the starting vertex. Below is a visual example. Your function should return TRUE for vertices 3, 4, or 5. It should return FALSE for 1 or 2. Your function should be named detectCycleHelper() and be called by detectCycle(), as shown below in the class definition.



Assume the following structure definitions-

```
struct vertex;

struct adjVertex{
    vertex *v;
};

struct vertex{
    int key;
    bool visited = false;
    int distance = 0;
    vertex *pred = NULL; // predecessor
```

```
        std::vector<adjVertex> adj;
};

class Graph
{
     private:
          std::vector<vertex*> vertices;
          vertex* search(int key); // no need to implement
          bool detectCycleHelper(vertex* v, int startK);


     public:
          bool detectCycle(int startK) {
          // Call the search function
          // Call the detectCycleHelper function
          }
    ...
          // assume all other methods are also implemented

};
```

NOTES:

1. The search method can be assumed to return the vertex reference corresponding with the key, no implementation is necessary.

2. Implement your solution in the detectCycle and detectCycleHelper methods. No other helper functions should be necessary. (**Hint:** You may want to implement your solution in the style of a depth first traversal.)

3. When you are ready to submit your solution, click the Canvas link named **Midterm II, Part II – Coding** under **Week 12** and upload all 3 of your solution files: graph.hpp, graph.cpp, and driver_graph.cpp.