CS302

Assignment 4: The Flash

Description

The flash. Known for his extraordinary speed. He can even go back in time and mess with the space time continuum. Which of course he does. In order to go back in time, he needs to run and pick up speed. The flash will begin his run, he needs to move from point A to point B but he wants to move in a zig zag pattern (so each point he can move forward that leans to the left or to the right), thus a binary tree will represent all the valid paths the flash can take. You job is to help the flash by computing the largest zig zag path in a binary tree that he can use to time travel and troll everyone in the DC Universe. You will need to implement the following class (this is the content of binTree.h)

```
#include <vector>
#include <string>

struct binTreeNode
{
  std::string location;
  binTreeNode * left;
  binTreeNode * right;
};

class binTree
{
public:
  binTree();
  ~binTree();
  void buildTree(std::vector<std::string>);
  std::vector<std::string> zigzag();
private:
  binTreeNode* buildTree(binTreeNode*, std::vector<std::string>, int);
  std::vector<std::string> zigzag(binTreeNode*, bool, std::vector<std::string>);
  void deallocateTree(binTreeNode*);

  binTreeNode * root;
};
```

You cannot modify this class or submit a different header file to code grade, here is the description of the members:

- `binTreeNode * root` - a pointer that points to the root of the binary tree
- `binTree::binTree()` - default constructor, sets `root = nullptr`
- `binTree::~binTree()` - destructor, only contains `deallocateTree(root)` in the body
- `void binTree::deallocateTree(binTreeNode * r)` - the function takes in a pointer of a node in the binary tree, and deallocates the tree rooted at r in a postorder fashion
- `void binTree::buildTree(std::vector<std::string> locations)` - a wrapper buildTree function, if the locations vector is empty then return, otherwise you would have

  ```
  root = buildTree(new binTreeNode(), locations, 0);
  ```

  This function would return the root of the tree and is then assigned to the `root` pointer, you pass in a node into the private buildTree function (that node will get processed in the private function body), you pass in the list of locations, and the index (in this case we pass in 0 since index 0 would be at the root node)
- `binTreeNode* binTree::buildTree(binTreeNode * r, std::vector<std::string> locations, int index)` - recursive function that builds a tree in preorder fashion, there are two base cases, if the index is beyond the bounds the locations vector or the element at index is an underscore, in those cases you want to return `nullptr` back to the parent (you may want to deallocate r because it will not be needed since you're returning a `nullptr`).

  In the general case you set `r->location = locations[index]`, then you need to call this function recursively twice (for each child), and assign `r->left` and `r->right` to what the recursive function returns (returns either a `nullptr` or an address of a node), thus each function call you pass in a `new binTreeNode()`, the locations vector, and the index of where the left/right child function will access, the left child would access index * 2 + 1 and the right child would access (index + 1) * 2, then at the end you return r and the parent of r would catch r.
- `std::vector<std::string> binTree::zigzag()` - a function that returns a vector of nodes of the longest zig zag path in the tree, if the tree is empty return an empty vector, this function initiates the zig zag finder (starting from the root), you need to call the private zigzag function on the root's left and right side (returns two vectors for a zigzag path found on the left and right), and then return the larger path
- `std::vector<std::string> binTree::zigzag(binTreeNode* r, bool childType, std::vector<std::string> path)` - a function that returns the longest path from node r (there are two possible zig zag paths for each node), if r is null then return an empty vector, else you have to get the zig zag path on the left and right side of r and return the larger path. The childType boolean denotes whether this child is a left child of its parent (`true`) or if it's a right child of its parent (`false`), the vector passed into this function is the partial zig zag path (which means if this parameter contains a path, the ancestor node(s) already determined that some zig zag exists)

Contents of Main

A main.cpp file is provided for you on canvas and will be used to link to your implementation file on code grade

Specifications

- Document your code

- Make sure your code is memory leak free
- No global variables

Example Output

```
% g++ main.cpp binTree.cpp
% ./a.out

Flash says Excelsior! Give the tree node names: A B C D E _ F _ _ G _ _ _ H
_ _ _ _ _ _ J _ _ _ _ _ _ _ _ _ _ -1

Longest Zig Zag path: A B E G J

%./a.out

Flash says Excelsior! Give the tree node names: A B C D _ E F _ G _ _ H _ I J
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ K _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ L _ _ -1

Longest Zig Zag path: F J K L
```

Submission

Submit binTree.cpp file to code grade by the deadline

References

- Link to the top image can be found at $https://www.deviantart.com/mahesh69a/art/Flash-Logo-Icon-254518134$
- Supplemental Video https://youtu.be/8zFG-6RM5Zo