Geoffrey Cramer
ECE368 – Summer 2017

Programming Assignment 3

Time Complexity:

First, a binary tree is read in from a text file with appropriate information. Additionally, I decided to fill in height and width dimensions for parent nodes (based on the its cut and its children's sizes) while creating the tree. Next, I did a pre-order traversal of the tree and updated x and y values along the way. This will be done with $O(n)$ time complexity because each node must be visited.

Space Complexity:

First, the structs used are listed below.

```
typedef struct treeNode {
  char cut;
  double x;
  double y;
  double height;
  double width;
  int ID; //identifying number (height if a parent node)
  struct treeNode* left;
  struct treeNode* right;
  struct treeNode* parent;
} treeNode;

typedef struct treeStack {
  treeNode* tree;
  struct treeStack* next;
} treeStack;
```

While reading in the file, a stack must be created. Each input will, at some point, be placed in the stack. Therefore, the stack itself has $O(n)$ space complexity. Each input will also have a place in the tree, meaning the tree itself also has a space complexity of $O(n)$. To perform the packing itself, a recursive approach was used. During the pre-order traversal, x- & y-coordinates are updated. The packing function must be called for each node in the tree, resulting in $O(n)$ space complexity for the algorithm itself.

Overall, my implementation has an $O(n)$ space complexity including the stack, tree and recursive packing algorithm.