Compilers Lab 5-6-7

Crafting a Compiler 9.2

```
signtest (exp) {
        neg : stmts
        zero : stmts
        pos : stmts
}
```

AST

<exp>
- <neg>
-- [stmts]

Exp is the start of the AST since it is the condition of signtest. Exp is the root since it is the start of the whole function. Then next is the sign, either neg, zero, or pos. One of these will be a branch from the root, since they are the next after the root, exp. The sign is the branch since it's going to be in the middle of the root and the leaf, or stmts. After the branch there is the stmts, these are the leaf since they are the end of signtest function. At the end that becomes the leaf since it is the end of the function

Crafting a Compiler 8.1

Binary Search Trees and Hash Maps are the most important things for semantic analysis. Binary Search Tree's (BST) are important to keep track of the different scopes that may take place when doing semantic analysis. BST's can keep track of the scopes because of the parent, left, and right scopes. There would have to be some changes from just not a left or right node since there can be more than two scopes inside of one scope itself. But the core idea is very helpful in order to keep track of the scopes for the semantic analysis. Then Hash Map's are helpful to keep the variables that are in the scope accounted for. We can add variables to the hashmap when they are found in the program. When we see the variable we add the variable and the type that it is assigned to. After that if we see the variable again in the scope then the hash map will clash. We can use this clash to give an error for semantic analysis about having an already declared variable. Hash maps also have the added bonus that the hash map's grow and do not have to be set to a defined amount.