Moti Begna

CSCI 5106

Homework 5

Problem 4 -

When considering the implementation of the binary search tree from problem 2, one of the advantages is that the use of unions allows for polymorphism while maintaining a consistent amount of allocated space in memory. Thus, the size for the variant part of the structure remains the same, regardless of what fields will be implemented. This would thus allow for great flexibility and extension for different data types. A glaring issue of this, however, is that this expression type lowers the security of the program, since fields that don't exist can still be accessed without any knowledge that it would be illegal since it cannot be checked statically. Thus, for example, if a union is defined to have a char* field and an int field, but only the int field is defined, the program would be freely able to access the char* field without any information as to why that would be an illegal access.

When considering the implementation of the binary search tree from problem 3, we can see that type safety can cause issues since explicit type declarations of void pointer types rely on the correct type being known prior to its usage. For example, (from my implementation) if one was to accidentally insert a string into a BST, but identify the insertion as using a person enum type, then access issues would arise where the program would assume that the quarried tree is of type person and try to access elements that don't exist. Even with this issue, however, the void pointer type allows for great flexibility in how values are parameterized since any element type can be used without needing to explicitly state it's type. Thus, the largest limitation to this implementation is correct type usage, such as knowing the element type prior to usage and correctly casting those elements during usage.