CSC 453 Milestone 9 (Offsets)

Goal:

This milestone is to decorate/annotate an Tau's variable and parameter symbols with their offsets. This phase directly follows typechecking.

Specifications

The Tau language specification is a separate document.

The AST nodes are defined in `asts.py`.

Symbols and types are defined in `symbols.py`.

Create `offsets.py`

You will start with the supplied stub file called `offsets.py` that will contain the code for annotating the AST and Symbols with computed offsets.

To accomplish this, you will walk the AST built by your parser (after doing typechecking).

Specifically you will need to: * Update the `offset` field of Symbols that represent local variables and parameters. Assume that all `int` and `bool` values have unit (`1`) size. * Update the `size` field of the `FuncDecl` AST node to be the total size of the local variablels **AND** the 3 extra slots for the return address, old frame pointer, and old stack pointer. (Note that the parameters are **not** included in the size of the local variables.) * The local variables must be *packed* (no gaps) as described in class. * Your compiler does not need to handle arrays.

NOTE: The parameter offsets will all be negative values relative to the `SP`/`FP`, but they will start at `-2`. (The return value will be at offset `-1`)

Example:

```
func f(
    int x,    // -2
    int y,    // -3
    int z)    // -4
{
    int a;    // 3
    {
        int b;    // 4
    }
    while true {
        int c;    // 4  <-- packed!!
        int d;    // 5
    }
}     // size = 6
```

Errors

This milestone only requires that you correctly annotate a correct Tau program.

Difficulty

This milestone does not require a lot of code beyond the tree walker, which is provided. That said, it can be a little tricky to check and infer types for some nodes.

Start early and ask questions.

Standard Requirements

Your program must meet all the requirements outlined in the common requirements document.