# Compiler 2 - Parser and Codegen for simple.jive

The next step in developing our compiler is to parse the tokens into an abstract syntax tree (AST) and traverse it for simple code generation. For now, don't worry about being able to handle the whole grammar. The only programs you need to deal with are of the form:

```
fn main() -> int
{
    return 42
}

fn foo() -> int
{
    return 64
}

fn  bar(  ) -> int{
    return 123
}

fn baz()->int{return 17}
```

That is any number of function definitions that have no parameters and return an int, whose body is a return statement with an integer literal as the return expression.

I've provided some starter code. It's filled with TODO's. Try to address them all. Also, definitely do read and try to understand the code that's there. Some things will have to be modified to work with your version of the lexer. You can also modify your lexer if you want.

Make sure you report an error to the user if there's something wrong with the input code. For instance, none of these should compile:

```
fn wrong() ->
{
    return 30
}

fn wrong2) -> int
{
    return 99
}

func wrong3() -> int
{
    return 78
}

fn 4() -> int
{
    return 4
}

fn wrong5() -> int
{
    return 55
```

Please submit all source files, including a build script, so that I can easily build and test your program. My recommended directory structure is as follows:

```
jive
    code
        main.c (https://northeastern.instructure.com/courses/225860/files/37244041?wrap=1) ↓
 (https://northeastern.instructure.com/courses/225860/files/37244041/download?download_frd=1)
        string.c (https://northeastern.instructure.com/courses/225860/files/37244058?wrap=1) ↓
 (https://northeastern.instructure.com/courses/225860/files/37244058/download?download_frd=1)
        lexer.c
        parser.c (https://northeastern.instructure.com/courses/225860/files/37244042?wrap=1) ↓
 (https://northeastern.instructure.com/courses/225860/files/37244042/download?download_frd=1)
        codegen.c (https://northeastern.instructure.com/courses/225860/files/37244040?wrap=1)
 ↓
 (https://northeastern.instructure.com/courses/225860/files/37244040/download?download_frd=1)
        build.sh (https://northeastern.instructure.com/courses/225860/files/37244039?wrap=1) ↓
 (https://northeastern.instructure.com/courses/225860/files/37244039/download?download_frd=1)
        // All C source code files live here
    jive_programs
        simple.jive (https://northeastern.instructure.com/courses/225860/files/37244043?wrap=1)
 ↓
 (https://northeastern.instructure.com/courses/225860/files/37244043/download?download_frd=1)
        simple2.jive (https://northeastern.instructure.com/courses/225860/files/37244044?wrap=
1) ↓
 (https://northeastern.instructure.com/courses/225860/files/37244044/download?download_frd=1)
        // All sample input programs for the compiler live here
    build
        // Output of the build process goes here (no need to submit this)
```

The above contains the links to the starter code. Good luck and happy coding. Please ask me any questions!

This tool needs to be loaded in a new browser window

Load Compiler 2 - Parser and Codegen for simple.jive in a new window