This is a declaration that this project has been solely my own work except elements which I have explicitly attributed to another source.

StudentName: Michael Wall
StudentNum: 13522003

For this module I had great difficulty applying the theory learned about abstract syntax trees, symbol tables and intermediate code representations to the work required for the assignment. I believe part of the issue was having to work forward from the code I wrote in the first assignment, along with a lack of understanding how jjtree is designed to be used.

I tested my program with the files `compilerTest1.txt` and `compilerTest2.txt` which were both included in my submission. In this document I only include output from running my program with `compilerTest1.txt`.

Abstract Syntax Tree

- Given the following input to my program:

```
var i:integer;

integer test_fn (x:integer, y:integer)
{
    var i:integer;
    i = 2;
    return (x);
}

void func ()
{
    return ();
}

main{
    // a simple comment
    /* a comment with /* nested /* comments */ of */ which  there
are several */

    var i: integer;

    i = 1;
    i = test_fn (i);
}
```

- I generated the following AST as output:

```
Abstract Syntax Tree:
```

```
PROG()
 VAR_DECL()
  ID(i)
  TYPE(integer)
 FUNC()
  TYPE(integer)
  ID(test_fn)
  PARAMLIST()
   ID(x)
   TYPE(integer)
   ID(y)
   TYPE(integer)
  VAR_DECL()
   ID(i)
   TYPE(integer)
  ASSIGN()
   ID(i)
   NUM(2)
  ID(x)
 FUNC()
  TYPE(void)
  ID(func)
 MAIN()
  VAR_DECL()
   ID(i)
   TYPE(integer)
  ASSIGN()
   ID(i)
   NUM(1)
  ID(i)
  ASSIGN()
   ID(test_fn)
   ID(i)
```

I was not sure which elements should be represented in the tree and how exactly they should be represented. I worked with the example given in the notes, but I feel like the example language was much simpler in structure and possible complexity than the one we were designated for this assignment. As a result I was unsure how to represent all of the possible scenarios in the AST.

Symbol Table

I generated my symbol table based on the examples given in the lecture notes. I researched scope in symbol tables, but again, I was unfortunately unable to apply the concept of scope to my symbol table.

- Given the same input as above, I generated the following symbol table:

```
Symbol Table:
2
 type = number
 value = 2
i
 type = integer
 value = i
1
 type = number
 value = 1
test_fn
 type = function
 value = test_fn
```

Semantic checking and intermediate code representation

Although I can understand pretty easily the theory of semantic checking and the theoretical process behind 3 address code generation, I was unable to understand what processes which were applied in the lecture notes example for jjtree, and I could not understand how to apply them to my code. This was also partly due to a lack of understanding at how the traversal of the AST worked in jjtree, and how to use the symbol table in my `PrintVisitor` to perform any meaningful checks. My best attempt at traversing the tree yielded the following output given the same input as above:

```
Program:
var i:integer;

integer test_fn   (x:integer, y:integer)
  var i:integer;
  i=2;
  x
void func
main {
  var i:integer;
  i=1;
  i  test_fn=i;
}
```

Conclusion

I am very disappointed in my performance in this assignment in contrast to my work and results in other modules (and in the first assignment for this module), especially considering the ratio of time I have invested in this assignment to the amount of possible marks I can obtain from it. As such, I cannot afford to invest any more time in this, so I would greatly appreciate any detailed feedback that you can give me on my submission.