CE305 Assignment Two

Small Compiler

The assignment is worth 25% of the total mark for this module. This assignment requires you to build a small compiler for a simple language (source language). The source language is open for you to design. The target language is Python language. The compiler should translate the source language into Python language. You can use ANTLR 4 and Java to build the compiler. Three pieces of sample code in target language (Python) are provided for demonstrating the main features of the source language. You need to provide three pieces of code in the source language. Then your compiler should be able to translate your code in the source language into the Python examples, which should be run correctly in Python.
Finally, you are required to submit a report which describes the specification of your source language, and the implementation of your developed compiler.

The Source Language

As part of your report on the assignment, you are expected to provide a specification of the source language. The design and implementation of the source language is open, but you should include

1. Basic notions
    - Variable declarations (e.g. int a)
    - Variable assignment statements (e.g. a:=b+1)
    - Expression (e.g. a:=(2+b*3)/(5+d))
    - Sequencing of statements (e.g. b:=2; a:=b+1)
2. Control flow structures
    - Conditional statements (e.g. if e=2 then f:=2, and if-then-else statements)
    - Unbounded iteration (e.g. while x<1 do x=x+1)
    - Statement blocks (so you can have more than one statement within the scope of a control flow statement, e.g. using begin, end, or {...}).

The Target Language

Python - you have to be able to demonstrate that your compiler is generating the Python code, and the Python code runs correctly and gives the correct result.

The Compiler

1. The basic system

You can use ANTLR 4 and Java to build the compiler. You have to select the parse tree listener interface or parse tree visitor interface in ANTLR to implement the translation. The compiler should be able to read a source file in your language, translate it into the Python language, and save the Python file into a text file. After the target text file is generated, you can run it in the Python environment to produce the correct result.

2. Additional features

Some marks will be awarded for additional features, including

- Abstract Syntax Tree
  Producing Abstract Syntax Trees (AST) that go beyond the built-in features of the ANTLR that produces the parse-tree.
- Error handling
  Producing error identification that goes beyond the built-in features of the development environment.

- Type checking
    - Building a symbol table and use it to report errors (such as undefined variables, or incorrect variable types).
    - Allowing local declaration of variables, not just global variables.
    - Allow declaration and use of floats and integers
- Functions

  Allow the declaration and use of functions with parameters and local variables. Note that this is advanced, especially if functions typing is supported.

The Python Sample Codes

Three sample codes in Python language are provided. These codes are for demonstrating the main features of the source language. For the additional features you should decide what code to demonstrate. Of course, these should include other aspect of your compiler such as error detection.

1. Fibonacci numbers (fibonacci.py)

```
n = 12
previous = 0
current = 1
index = 1
while index <= n:
  print(current)
  temp = current
  current = current + previous
  previous = temp
  index += 1
```

This code should produce these numbers vertically: 1 1 2 3 5 8 13 21 34 55 89 144

2. Factorial (factorial.py)

```
n=8
val = 1
counter = 1
while counter <= n:
  val = val * counter
  counter = counter + 1
  print(val)
```

This code should produce: 40320

3. Odd, Even and Sum (odd_even.py)

```
n = 100
odd = 0
even = 0
index = 1
while index <= n:
 if index % 2 == 0:
        even = even + index
 else:
        odd = odd + index
 index = index + 1
 print(odd)
 print(even)
 print(odd + even)
```

This code should produce these numbers vertically: 2500 2550 5050

The Report
The report should contain the specification and description of the implementation. Your report must include some explanatory narrative, describing what has been implemented, and how, highlighting particular features, or shortcomings. The report should be in PDF and must include a cover page with the module name, student name and ID. The source code (for the node you developed) should be made as a zip file and submitted together with the report. The maximum length of the report is of 10 pages (include the appendix).

The report should include

- A section that describes the source language.
  You are required to produce and present details of the formal specification of the source language including the regular expressions and BNF grammar that is implemented by the parse.
- A section that describes the compiler.
  You are required to present details of the implementation of the compiler, and the description of how the tools were applied.

Marking Scheme
Your assignment marks are broken down into the following:

- 50% for the design of the source language
  - Definition of Tokens (10%)
  - Grammar of Expressions (15%)
  - Description of Translation (15%)
  - Extended features (10%)
- 40% for the implementation of the complier including extended features
  - Basic operations (20%)
  - Coding quality and/or use of tools (10%)
  - Extended features (10%)
- 10% for clarity of the report.

Submission:
The report should be submitted through the online coursework submission system (FASER). You should submit a single pdf file and a zip file.

Late submission and plagiarism
This assignment is to be done individually, i.e. whatever you submit must be your own individual work. Any software, figures, or any other materials that you use in this assignment, whether previously published or not, must be referred to and properly acknowledged. Please be aware that the module supervisor may ask students for an interview to explain their submitted work.