Fundamentals of Programming Languages

Task Two - referral work

Introduction

In this assignment you must implement a compiler and use it to convert programs written using the language that you defined in the first assignment into JavaScript.

This assignment is worth 60% of the module's marks.

The task

In this assignment you will write a compiler for the PDL language. You can find the grammar on Blackboard in both the original assessment folder and here in the assessment folder.

You may use either the visitor or listener implementations. Each will be marked in the same way and to the same standards.

1. Using ANTLR and Java, write an application which builds a parse tree for valid programs. You should write code which demonstrates that the tree can be walked.
2. Extend your application so that a user can search the parse tree for language elements or grammar rules.
3. Transform simple programs into JavaScript.

Two sample grammars are available on the Blackboard site. If you feel that your own grammar lacks sufficient richness to be used in tackling this problem then you may use one of the provided solutions instead. You must, though, write your own sample programs.

Learning outcomes

This assignment partially covers the following learning outcomes:

- Understand the principles which lie beneath programming language design,
- Design formal representations of language constructs,
- Implement the basic components of a simple compiler.

Submission

You must submit a single zip file containing all of your code plus instructions for using it and sample programs through the SHU assignment handler on Blackboard.

The deadline is **3 p.m.** on **Thursday 8th July, 2021**.

Marking Scheme

Your work will be marked using a grade-based approach that is described in a separate document available on the module’s Blackboard site and at http://tinyurl.com/zkej95m.

The following table is an incomplete list of the types of thing that you should demonstrate in your video.

Aspect	Marks available	At pass level	At distinctive level
Displaying the parse tree	25	● Some attempt is made to print the tree. This may be as a simple string. ● Very simple trees can be output	● Trees are displayed successfully for most valid, and parsable, programs. ● The output may be a useful intermediate representation such as ○ S-expressions ○ JSON documents ○ XML structures
Searching of the parse tree	25	Some attempt is made which may ● accept a search criterion ● build an appropriate data structure to hold the result ● start to search the tree ● attempt to return a result	● Search criteria can be entered and chained ● Most valid programs can be searched ● Searches return results where those results are available ● Results are presented clearly and usefully ● No matching data is still treated as a valid result
Transform a program into JavaScript	50	Some attempt is made to produce output. This may include ○ creation of suitable templates ○ mapping of templates to rules ○ modification of the grammar to support the necessary set of rules	● A range of valid programs can be transformed into valid outputs that can be executed. ● Code is clear and appropriate ● Correct structures are used ● Errors or edge-conditions are handled gracefully