Project Assignment

Description

DCC053 (Compiler Construction) is a project-oriented course where students will implement two distinct methods for executing a subset of Standard ML of New Jersey, a functional programming language. The first method is a tree-based interpreter, and the second is a code generator that produces RISC-V instructions. These methods will differ in their capabilities: the interpreter will handle closures and recursive functions, while the code generator will manage anonymous functions but not recursive functions. By the end of the project, students will be able to interpret functions like the examples shown below:

Computes the factorial of a number, and tests it with factorial of 10	**Computes the n-th term of the Fibonacci sequence, and tests it with the 5th term**

`let`
`  fun fact n =`
`  if n < 2`
`  then 1`
`  else n * fact (n-1)`
`in`
`  fact 10`
`end`
	`let`
	`  fun fib n =`
	`  if n < 2`
	`  then 1`
	`  else fib (n-1) + fib (n-2)`
	`in`
	`  fib 5`
	`end`

Count the digits of an integer number, and tests it with "123456"	**Sums up the digits of an integer number, and tests it with "123"**

`let`
`  fun countDigits n =`
`  if n = 0`
`  then 0`
`  else 1 + countDigits (n div 10)`
`in`
`  countDigits 123456`
`end`
	`let`
	`  fun sDigs n =`
	`  if n = 0`
	`  then 0`
	`  else (n mod 10) + sDigs (n div 10)`
	`in`
	`  sumOfDigits 123`
	`end`

The project will be implemented in Python and graded through UFMG's Moodle system. It consists of 15 deliverables, each comprising a set of Python programs that together provide some functionality of our "MiniSML." There are dependencies between deliverables; however, the longest chain of dependencies involves only eight deliverables. This means students can complete substantial portions of the project even if some earlier parts are unfinished. Each deliverable includes partially completed code that students must complete. While students have the freedom to choose how to implement the missing sections, they must work within the provided Python modules and cannot remove or add new files. Some sub-projects will require modifications to earlier deliverables, allowing students to practice skills such as grammar refactoring. For example, the precedence of the unary "not" operator will evolve over the course of the project, and the syntax for declaring variables will also change.

The different parts of the project are represented by the following acyclic graph:

Educational Goals

The compiler construction project for DCC053 is designed to provide students with a comprehensive, hands-on learning experience in both the theory and practice of compiler development. Over the course of 15 weeks, students will progressively build a compiler for a subset of Standard ML of New Jersey in order to achieve the following educational goals:

1. **Understanding Compiler Theory**:
    - **Type Checking and Type Inference**: Students will learn the foundational concepts of type checking and type inference, critical for ensuring the correctness of program computations.
    - **Syntax and Semantics**: Students will gain an understanding of syntax and semantic analysis, including how to parse and interpret code according to the language's rules.

2. **Learning Compiler Engineering**:
    - **Program Representation**: Students will learn how to represent programs internally using abstract syntax trees (ASTs) and other data structures that facilitate analysis and transformation.
    - **Code Generation**: Students will develop skills in translating high-level language constructs into low-level instructions, specifically RISC-V assembly code.
    - **Register Allocation**: Students will understand the complexities of register allocation, a key aspect of optimizing the use of a processor's registers during code generation.
    - **Recursive Functions and Closures**: Students will implement support for advanced programming constructs such as recursive functions and closures, learning about the challenges and solutions associated with these features.
3. **Applying and Reinforcing Concepts**:
    - **Gradual Development**: By working on weekly deliverables, students will apply the concepts learned in lectures in a practical setting, reinforcing their understanding through continuous, iterative development.
    - **Incremental Changes**: Students will experience and adapt to changes in implementation decisions, such as modifying operator precedence, mimicking real-world scenarios where compilers evolve and require adjustments.
4. **Practical Experience**:
    - **Iterative Design**: Students will engage in iterative design and development, gaining experience in modifying and improving their compiler as new features are added.
    - **Debugging and Optimization**: Students will develop debugging and optimization skills, essential for refining compiler performance and correctness.

By working through these deliverables, students will not only acquire a solid theoretical foundation in compiler construction but also practical experience in building a functional compiler. The project is structured to ensure that students continuously engage with and apply the concepts they learn in class, providing a thorough and integrated understanding of compiler design and implementation.

How to Approach the Project

Each deliverable in this project consists of several Python files that you will need to work on. To test your implementation locally, you can use doctests. For example, to test the file `Expression.py`, you would use the following command:

```
python3 -m doctest Expression.py
```

Initially, your tests will fail because the code hasn't been implemented yet. However, this is expected! Begin by carefully reading the tests and working on the implementation incrementally. Your goal is to address each test case, gradually refining your code until all the tests pass for that particular deliverable.

The project will be graded through Moodle, and you are allowed to submit your work as many times as needed. Moodle will provide you with a comprehensive set of tests for each deliverable. If some of your tests fail, don’t be discouraged. You can either address the issues identified by the failing tests or focus on other parts of the project. Remember that some tests are weighted less heavily, so a few failures may not significantly impact your overall grade.

To effectively manage your work, try to follow a test-driven development approach. The tests are already provided for you, so use them as a guide to ensure your implementation meets the required functionality. By doing so, you will systematically address the requirements and incrementally build a robust solution.

Approach each deliverable with a step-by-step mindset, using the tests to drive your development. This method will help you stay focused and organized, making the project more manageable and less overwhelming.

And don't feel overwhelmed...

Embarking on the journey of building a compiler through this project may seem daunting at first, but remember, you are not alone, and you have the tools and support needed to succeed. The project is designed to be manageable and engaging, with each of the 15 weekly deliverables being small and focused on specific tasks. Most of the code and concepts required will be thoroughly covered in class, so you'll have a solid foundation to build upon.

Don't hesitate to consult resources like LLMs (e.g., ChatGPT) for guidance and clarification—they're here to help you understand and overcome challenges. However, it’s important that the work you submit is your own, as this will ensure you gain the most from the project and truly develop your skills.

Each deliverable is accompanied by a set of tests that will help you verify your work and guide you through the development process. Even if you don’t pass all the tests on your first attempt, partial credit is available, and you can always improve your solutions. The project is structured in such a way that even if some deliverables are not completed on time, you can continue to work on other parts of the project. The deliverables are organized in a partially ordered set, so you can still make significant progress.

Approach each task step by step, and remember that every small victory contributes to your overall success. The skills you gain from this project will be invaluable, both academically and professionally. Embrace the challenge, stay persistent, and take pride in the progress you make. You're building something complex and powerful, and you have the capability to achieve it. Keep pushing forward, and you'll see how your hard work pays off!