Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Assessed Exercise 2, Task 1

Implement the part of the language in black. No need to implement code generation for the red parts. No test for this task will feature programs containing language constructs in red. Your code generator can throw `CodegenException` for the ASTs that correspond to language constructs in red, or you can return MIPS code, it does not matter.

```
PROG → DEC | DEC; PROG
DEC → def ID (VARDEC) = E
VARDEC →  ε | VARDECNE
VARDECNE → ID | VARDECNE, ID
ID → ... (identifiers)
INT → ... (Integers)
E →  INT
   | ID
   | if E COMP E then E else E endif
   | (E BINOP E)
   | (E)
   | ID(ARGS)
   | skip
   | (E; E)
   | while E COMP E do E endwhile
   | repeat E until E COMP E endrepeat
   | ID := E
   | break
   | continue
ARGS → ε | ARGSNE
ARGSNE → E | ARGSNE, E
COMP → == | < | > | <= | >=
BINOP → + | - | * | /
```

Assignment Project Exam Help
https://powcoder.com

Recall that the relevant definitions are here, here and here. If you don't want to implement a feature, simply throw `CodegenException` when the code generator encounters this feature.

Add WeChat powcoder

Note that the translation of procedure invocation (given by the production `E → ID(ARGS)` in the grammar above) is part of Task 1. Since every program will have at least one procedure invocation, I strongly suggest that you focus on getting this right.