6CCS3CFL-CW3

Q1

Grammer:

For making my parser recognise tokens, I had to change the implementation of a few essential functions.

I introduced a few new implicit definitions which correspond to   
Parser[List[Token], Token]  
And also introduced some atomic parsers for some specific tokens.

Boolen Expressions

Graphical user interface

Description automatically generated

Arithmetic Expressions

A screenshot of a computer

Description automatically generated with medium confidence

Single Statements

Had to introduce extra cases of write since there were various different ways it was implemented in test programs.

A screenshot of a computer

Description automatically generated with medium confidence

Compound Statements

A screenshot of a computer

Description automatically generated with medium confidence

Blocks

Text

Description automatically generated

Q2

Output of running the respective programs:

\_\_\_\_\_\_\_\_\_\_Fib \_\_\_\_\_\_\_\_\_\_

HashSet(List(WriteStr("Fib"), Read(n), Assign(minus1,Num(0)), Assign(minus2,Num(1)), While(Bop(>,Var(n),Num(0)),List(Assign(temp,Var(minus2)), Assign(minus2,Aop(+,Var(minus1),Var(minus2))), Assign(minus1,Var(temp)), Assign(n,Aop(-,Var(n),Num(1))))), WriteStr("Result"), WriteVar(minus2)))

\_\_\_\_\_\_\_\_\_\_Loops \_\_\_\_\_\_\_\_\_\_

HashSet(List(Assign(start,Num(100)), Assign(x,Var(start)), Assign(y,Var(start)), Assign(z,Var(start)), While(Bop(<,Num(0),Var(x)),List(While(Bop(<,Num(0),Var(y)),List(While(Bop(<,Num(0),Var(z)),List(Assign(z,Aop(-,Var(z),Num(1))))), Assign(z,Var(start)), Assign(y,Aop(-,Var(y),Num(1))))), Assign(y,Var(start)), Assign(x,Aop(-,Var(x),Num(1)))))))

\_\_\_\_\_\_\_\_\_\_Primes \_\_\_\_\_\_\_\_\_\_

Set(List(Assign(end,Num(100)), Assign(n,Num(2)), While(Bop(<,Var(n),Var(end)),List(Assign(f,Num(2)), Assign(tmp,Num(0)), While(And(Bop(<,Var(f),Aop(+,Aop(/,Var(n),Num(2)),Num(1))),Bop(==,Var(tmp),Num(0))),List(If(Bop(==,Aop(\*,Aop(/,Var(n),Var(f)),Var(f)),Var(n)),List(Assign(tmp,Num(1))),List(Skip)), Assign(f,Aop(+,Var(f),Num(1))))), If(Bop(==,Var(tmp),Num(0)),List(WriteVar(n)),List(Skip)), Assign(n,Aop(+,Var(n),Num(1)))))))

\_\_\_\_\_\_\_\_\_\_Collatz \_\_\_\_\_\_\_\_\_\_

Set(List(Assign(bnd,Num(1)), While(Bop(<,Var(bnd),Num(101)),List(WriteVar(bnd), WriteStr(": "), Assign(n,Var(bnd)), Assign(cnt,Num(0)), While(Bop(>,Var(n),Num(1)),List(WriteVar(n), WriteStr(","), If(Bop(==,Aop(%,Var(n),Num(2)),Num(0)),List(Assign(n,Aop(/,Var(n),Num(2)))),List(Assign(n,Aop(+,Aop(\*,Num(3),Var(n)),Num(1))))), Assign(cnt,Aop(+,Var(cnt),Num(1))))), WriteStr(" => "), WriteVar(cnt), WriteStr("\n"), Assign(bnd,Aop(+,Var(bnd),Num(1)))))))

The tokenised output has \n and quotes since my implementation of the program manipulates strings in the eval\_stmt function.

Output for (**if** (a **<** b) **then skip else** a :**=** a \* b + 1)

Set(List(If(Bop(<,Var(a),Var(b)),List(Skip),List(Assign(a,Aop(+,Aop(\*,Var(a),Var(b)),Num(1)))))))

Parse Tree for (**if** (a **<** b) **then skip else** a :**=** a \* b + 1) is denoted below.

Diagram

Description automatically generated

Q3

The Time measurements are outlined below.

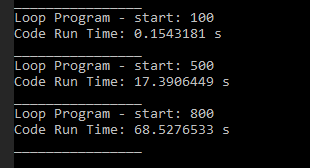
1. **Fibonacci**



The formatting of the code is not ideal since there are no newlines in the actual code of the program.

The write function doesn’t use **println** instead it uses **print**.

1. **Three Nested Loops**



Started off with 100 which took less than half a second.  
Then 500 took 18 seconds to execute and lastly having a start value of 800 took just a little more than 1 minute.

1. **Factors**

Shape

Description automatically generated with medium confidence

The formatting of this code is better since when the screenshot was taken, the write function was using **println** to print output. But since it is now using **print**, the new output is denoted below:



1. **Collatz**

