README

Test the program

If you'd like to test your code to see if your parse function and interpreter function work, start GHCI with `stack repl` as the follows:

```
$ stack repl
... More Out Put ...
ghci> :l monadic_interpreter.hs
Ok, one module loaded.

ghci> parse com s1String
[(Declare "x" (Constant 150) (Declare "y" (Constant 200) (Seq
(While (Greater (Variable "y") (Constant 0)) (Cond (Greater
(Variable "x") (Constant 100)) (Assign "x" (Divide (Variable "x")
(Constant 2))) (Assign "y" (Minus (Variable "y") (Constant 100)))))
(Print (Variable "x")))),"")]

ghci> unStOut(interpret1 s1 [])[]
((),[],"75")
```

We can see that with `parse com s1String`, it built a AST that is the same as the s1 we create, which means the parse function is working.

With `unStOut (interpret1 s1 []) []`, we get the result `((),[],"75")`. 75 is the final value of `x` of the program, which means that the interpreter function is working.

We can also test the language we just built as the follows:

```
ghci> main
>declare x = 150 in declare y = 200 in {while y > 0 do if x > 100
then x:=x/2 else y:=y-100; print x}
"75"
```

The answer is correct, which means the language works.