Bonuses

REPL

Rigor

Syntax Documentation for compiled code

Arrays

We implemented syntactic sugar for LISP style lists so that they may be written using a more familiar style

`(1 2 3 4 5)` -> `[1, 2, 3, 4, 5]`

We chose syntactic sugar over native handling as it will result in less written code in project overall.

Arrays can contain any amount of Atomic values.

Array Nesting

Arrays can have an infinite amount of nesting, to represent 2d arrays, 3d arrays and so on.

```
[[1,2,3,4,5],[1,2,3,4,5]]
```

Array Operation

There are a total of 7 array operations:

Head

`head` returns the first element of an array.

```
> (define a [1, 2, 3])
> (head a)
1
```

Last

`last` returns the first element of an array.

```
> (define a [1, 2, 3])
> (last a)
3
```

Tail

`tail` returns the array with its first element removed.

```
> (define a [1, 2, 3])
> (tail a)
[2,3]
```

Init

`init` returns the array with its last element removed.

```
> (define a [1, 2, 3])
> (init a)
[1,2]
```

Join

`join` concatenates 2 arrays and returns it.

```
> (define a [1, 2, 3])
> (define b [4, 5, 6])
> (join a b)
[1,2,3,4,5,6]
```

PushFront

`pushFront` adds an element to the beginning of an array and returns the array.

```
> (define a [2, 3, 4])
> (define b 1)
> (pushFront a b)
[1,2,3,4]
```

Pushback

`pushBack` adds an element to the end of an array and returns the array.

```
> (define a [1, 2, 3])
> (define b 4)
> (pushBack a b)
[1,2,3,4]
```

Strings and character escaping

It is possible to use strings in our code, they are written using the following syntax : `"Hello World!"`. It is also possible to escape the character `"` to use it inside of strings.

```
> (print "\"Hello World!\"")
"Hello World!"
```

Input/Output

Reading from Stdin

Writing to Stdout

Reading files

Floating point numbers

Our language supports **floating point numbers** as well as any **corresponding basic arithmetic operation** and full array support.

```
> (3.2 + 4.5)
7.7
```

```
> (3.2 - 4.5)
-1.3
```

```
> (3.2 * 4.5)
14.400001
```

```
> (3.2 / 4.5)
0.7111111
```

Rigor

We would also like to take the time to point out a couple best practices that we implemented rigorously while working on this project.

Here are a couple of these points :

- Issue tracking - **Over 60 issues**
- Pull request workflow - **Over 90 pull requests**
- Concise and meaningful commits - **Over 500 commits**
- Commit linting using commitizen
- Lots of code commenting

We also added **an other BNF style** document along with some documentation for our custom assembly like language that we compile into.