Vlang (Compiles to C or C++)

Statements:

Vlang statements are placed on a single line and end in ';'

Data types:

scl -- Integer scalar

vec -- Vector of scalars

Constants/Literals:

scl -- as for ***int*** in C

vec -- [scl_1, scl_2, ..., scl_n]

Variable definition:

scl s;

vec v{<sz>};

Operators:

. | **:** - Highest precedence left associative

+ | **-** | ***** | **/** | - Associativity/Precedence as in C

(...) -- Parenthesis (as in C)

Expressions:

scl – as in C for ***int*** (but without **++** / **--** and without shortcuts **+=** ***=** etc.)

vec expressions:

<vec> = <vec constant>, where constant size must equal <vec> size

<vec> = <vec variable>, where <vec variable> size must equal <vec> size

<vec> = <scl> -- all <vec> elements are assigned the scalar value

<vec> {+|-|*|/} <scl> -- Add/subtract/mult/divide <scl> on each $<vec_i>$

$<vec^l>$ {+|-|*|/} $<vec^r>$ -- Add/subtract/mult/divide for each $<vec^l>_i$ $<vec^r>_i$ – sizes must be equal

$<vec^l>$.$<vec^r>$ -- Vector dot-product ➔ ***scalar***: $\sum_i vec^l{}_i * vec^r{}_i$ – sizes must be equal

Recursive:

<exp> can be **`<exp> {+|-|*|/} <exp>`**

<exp> can be **`(<exp>)`**

Indexing:

<vec>:<scl> -- ➔***scalar:*** <vec> element at position <scl> (zero-based)

$<vec^l>$:$<vec^r>$ -- ➔***vector:*** <vec> composed of $<vec^l>$ elements indexed by $<vec^r>$ elements
<vec>:i = $<vec^l>$:($<vec^r>$:i), …] for all i – sizes must be equal
For example: [2, 4, 6, 8]:[1, 1, 3, 2] ➔ [4, 4, 8, 6]

Statement:

`<exp>;`

Conditional:

`if` <scl> {
  <stmnt>… -- do these statements if <scl> != 0 (i.e., not false)
}

Loops:

`loop` <scl> {
  <stmnt>… -- do these statements <scl> times
}

Printing:

`print` *element*[,*element* …];

element can be: <scl> -- prints scalar, as "%d\n" in C

element can be: <vec> -- prints vector as "**[**%d, %d, …,%d**]**\n" in C (for all vector elements)

Example:

```
scl x;
scl y;
scl i;
vec v1{6};
vec v2{6};
vec v3{6};
x = 2;
v1 = 2*x;              //v1 should now be [4,4,4,4,4,4]
v2 = [1,1,2,2,3,3]
print v1.v2            //should print: 48
y = v2:4;              //y = 3
i = 0;
loop y {
      v1:i = i;
      i = i + 1;
      }
print v2:v1                        //should print: [1,1,2,3,3,3]
print v2:v1:[5,4,3,2,1,0]          //should print: [3,3,3,2,1,1]
v3 = v1+v2                         //v3 should be [1,2,4,6,7,7]
print v2:([2,1,0,2,2,0].v3/10)     //should print: 2
```

```
vec a {3};
a = [10, 0, 20];
i=0;
loop 3 {
      if a.[1, 0, 0] {
            print i, a;      //0: [10, 0, 20]
                             //1: [20, 10, 0]
            a = a:[2, 0, 1];      //This rotates a to the right
            }
      i = i+1;
      }
vec z {4};
z = 10;
z = (z + [2, 4, 6, 8]) / 2;      //[6, 7, 8, 9]
z = z – 3 + [2, 3, 4, 5];
print z;                         //Prints: [5, 7, 9, 11]
print z.[1, 1, 1, 1];            //Prints: 32
```