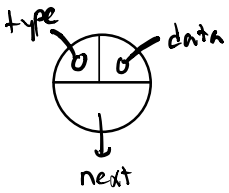
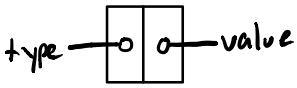
AST_NODE



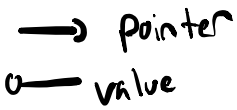

```
typedef enum ast_node_type {
    NUM_NODE_TYPE,
    FUNC_NODE_TYPE
} AST_NODE_TYPE;
```

```
typedef struct ast_node {
    AST_NODE_TYPE type;
    union {
        AST_NUMBER number;
        AST_FUNCTION function;
    } data;
    struct ast_node *next;
} AST_NODE;
```

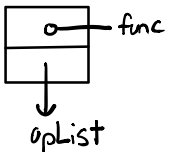
AST_NUMBER




```
typedef struct {
    NUM_TYPE type;
    double value;
} AST_NUMBER;

typedef AST_NUMBER RET_VAL;
```

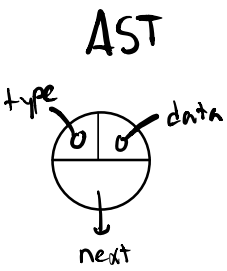


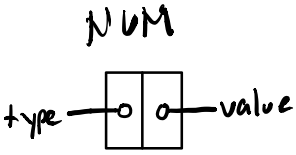
AST_FUNCTION

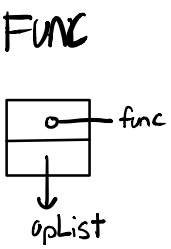


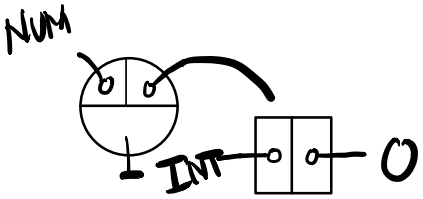

```
typedef struct ast_function {
    FUNC_TYPE func;
    struct ast_node *opList;
} AST_FUNCTION;
```

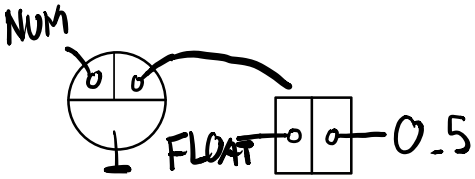
```
program ::= s_expr EOL | s_expr EOFT | EOL | EOFT

s_expr ::= f_expr | number | QUIT

f_expr ::= ( FUNC s_expr_section )

s_expr_section ::= s_expr_list | <empty>

s_expr_list ::= s_expr | s_expr s_expr_list

FUNC ::= neg | abs | add | sub |
        mult | div | remainder | exp |
        exp2 | pow | log | sqrt |
        cbrt | hypot | max | min

number ::= INT | DOUBLE

INT ::= optional +/-,
        then some digits

DOUBLE ::= optional +/-,
        then some digits,
        then a decimal point,
        then optionally some more digits

QUIT ::= quit
```


AST
type
data
next


NUM
type
value

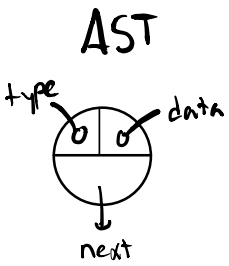

FUNC
func
opList

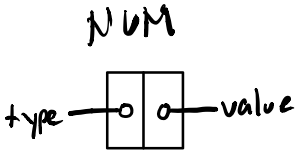

NUM
INT
0

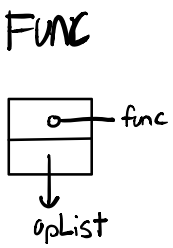

NUM
FLOAT
0.5

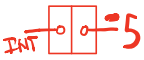
0

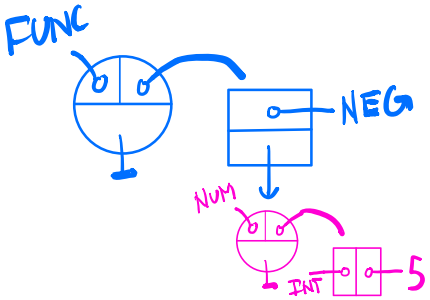
0.5


AST
type
data
next

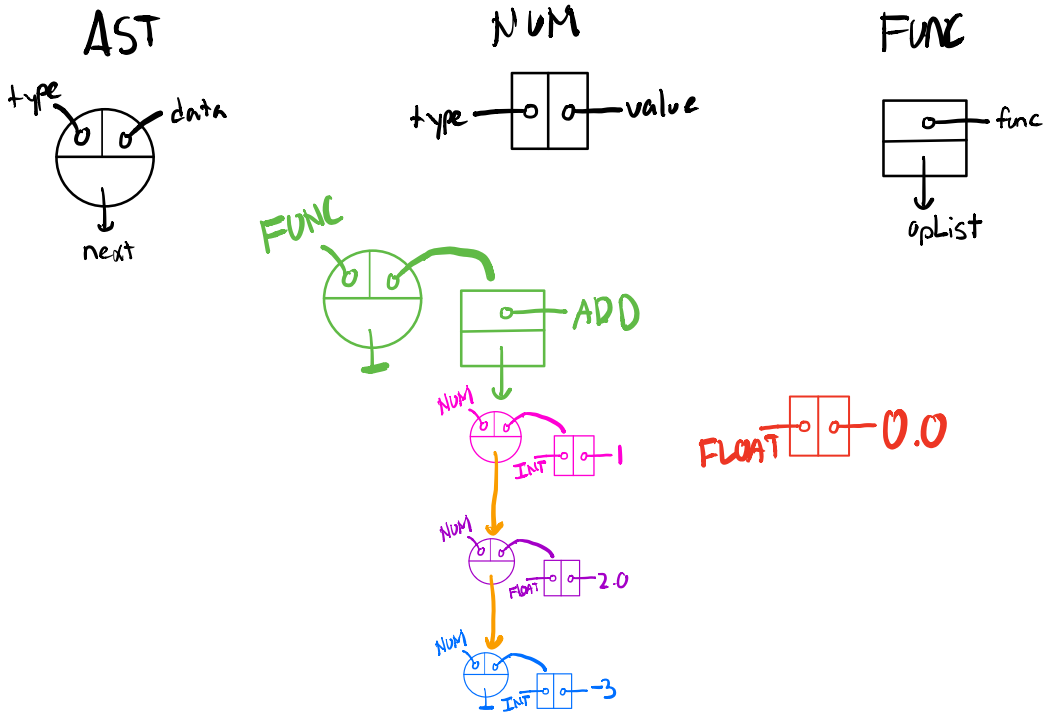

NUM
type
value


FUNC
func
oplist

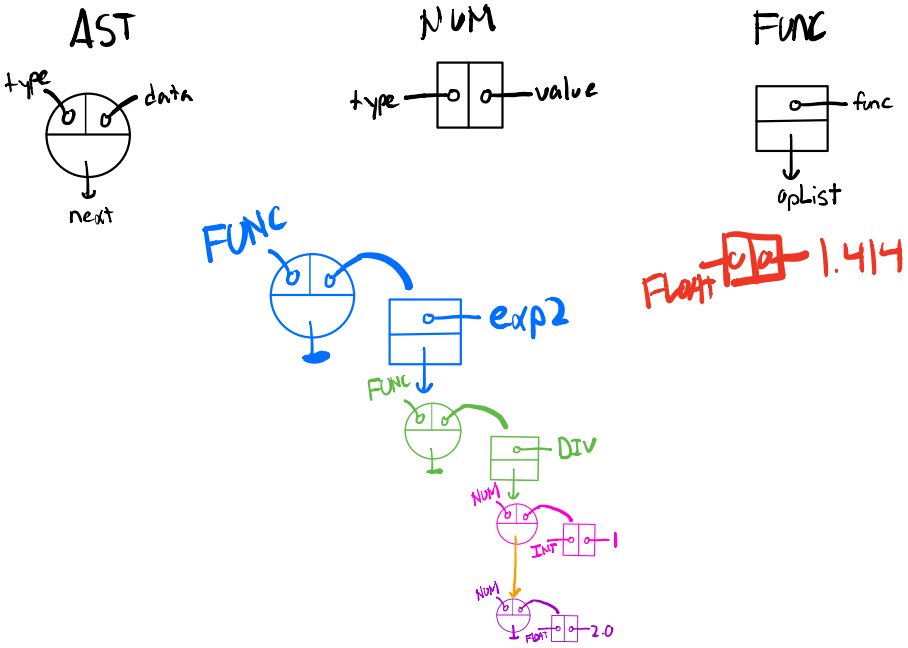

INT
5


FUNC
NEG
NUM
INT
5

(neg 5)


AST
type
data
next
NUM
type
value
FUNC
func
oplist
FUNC
ADD
NUM
INT
1
NUM
FLOAT
2.0
NUM
INT
-3
FLOAT
0.0

(add 1 2.0 -3)


AST
type
data
next
NUM
type
value
FUNC
func
oplist
FUNC
exp2
FUNC
DIV
NUM
INT
1
NUM
FLOAT
2.0
FLOAT
1.414

(exp2 (div 1 2.0))