Project – Part 1 (Scanner)

Execution examples:

```
/* Loop: for, while; if */
```

```
/* Example #1 */
    while (dec != 0) {
        oct += (dec % 8) * i;
        dec /= 8;
        i *= 10;
    }
    return oct;
```

```
IT'S A COMMENT : /* Loop: for, while; if */

IT'S A COMMENT : /* Example #1 */

WHILE : while
LEFT_PAREN : (
IDENTIFIER : dec
RELATIONAL OPERATOR : !=
INT_CONST : 0
RIGHT_PAREN : )
LEFT_CURL : {

IDENTIFIER : oct
ASSIGNMENT OPERATOR : +=
LEFT_PAREN : (
IDENTIFIER : dec
PERCENT : %
INT_CONST : 8
RIGHT_PAREN : )
STAR : *
IDENTIFIER : i
SEMICOLON : ;

IDENTIFIER : dec
ASSIGNMENT OPERATOR : /=
INT_CONST : 8
SEMICOLON : ;

IDENTIFIER : i
ASSIGNMENT OPERATOR : *=
INT_CONST : 10
SEMICOLON : ;

RIGHT_CURL : }

RETURN : return
IDENTIFIER : oct
SEMICOLON : ;
```

```c
/* Example #2 */
int i;
char temp[MAX_NAME + MAX_NUMBER];
if ((F_tel = fopen(File, "r")) == NULL) return 1;
```

```
IT'S A COMMENT : /* Example #2 */

INT : int
IDENTIFIER : i
SEMICOLON : ;

CHAR : char
IDENTIFIER : temp
RIGHT_SQRB : [
IDENTIFIER : MAX_NAME
PLUS : +
IDENTIFIER : MAX_NUMBER
LEFT_SQRB : ]
SEMICOLON : ;

IF : if
LEFT_PAREN : (
LEFT_PAREN : (
IDENTIFIER : F_tel
ASSIGN : =
IDENTIFIER : fopen
LEFT_PAREN : (
IDENTIFIER : File
COMMA : ,
STRING : "r"
RIGHT_PAREN : )
RIGHT_PAREN : )
RELATIONAL OPERATOR : ==
IDENTIFIER : NULL
RIGHT_PAREN : )
RETURN : return
INT_CONST : 1
SEMICOLON : ;
```

```
/* Example #3 */
for(int i=0; i<10; i++)
  {
    for (int j = 0; j < 10; j++)
    {
      if (j > i)
        break;
    }
  }
```

```
IT'S A COMMENT : /* Example #3 */

FOR : for
LEFT_PAREN : (
INT : int
IDENTIFIER : i
ASSIGN : =
INT_CONST : 0
SEMICOLON : ;
IDENTIFIER : i
RELATIONAL OPERATOR : <
INT_CONST : 10
SEMICOLON : ;
IDENTIFIER : i
INCREMENT : ++
RIGHT_PAREN : )

LEFT_CURL : {

FOR : for
LEFT_PAREN : (
INT : int
IDENTIFIER : j
ASSIGN : =
INT_CONST : 0
SEMICOLON : ;
IDENTIFIER : j
RELATIONAL OPERATOR : <
INT_CONST : 10
SEMICOLON : ;
IDENTIFIER : j
INCREMENT : ++
RIGHT_PAREN : )

LEFT_CURL : {

IF : if
LEFT_PAREN : (
IDENTIFIER : j
RELATIONAL OPERATOR : >
IDENTIFIER : i
RIGHT_PAREN : )

BREAK : break
SEMICOLON : ;

RIGHT_CURL : }

RIGHT_CURL : }
```

```
/* Different Types */

int b = 2147483647;
float r = 1.89;
long double n = r*123;
unsigned a = 4294967;
char g = 'Q', k = 'm';
const long int k = 25;
const double A=2.128E-2;

int a[2][3];
float arr[100];
char ch[3] = {'d', 'e', '9'};
struct { double x,y; } s1, s2, sm[9];
st1.name="Bibi";
```

```
IT'S A COMMENT : /* Different Types */

INT : int
IDENTIFIER : b
ASSIGN : =
INT_CONST : 2147483647
SEMICOLON : ;

FLOAT : float
IDENTIFIER : r
ASSIGN : =
FLOAT_CONST : 1.890000
SEMICOLON : ;

LONG : long
DOUBLE : double
IDENTIFIER : n
ASSIGN : =
IDENTIFIER : r
STAR : *
INT_CONST : 123
SEMICOLON : ;

UNSIGNED : unsigned
IDENTIFIER : a
ASSIGN : =
INT_CONST : 4294967
SEMICOLON : ;

CHAR : char
IDENTIFIER : g
ASSIGN : =
CHAR : 'Q'
COMMA : ,
IDENTIFIER : k
ASSIGN : =
CHAR : 'm'
SEMICOLON : ;
```

```
CONST : const
LONG : long
INT : int
IDENTIFIER : k
ASSIGN : =
INT_CONST : 25
SEMICOLON : ;

CONST : const
DOUBLE : double
IDENTIFIER : A
ASSIGN : =
FLOAT_CONST : 0.021280
SEMICOLON : ;


INT : int
IDENTIFIER : a
RIGHT_SQRB : [
INT_CONST : 2
LEFT_SQRB : ]
RIGHT_SQRB : [
INT_CONST : 3
LEFT_SQRB : ]
SEMICOLON : ;

FLOAT : float
IDENTIFIER : arr
RIGHT_SQRB : [
INT_CONST : 100
LEFT_SQRB : ]
SEMICOLON : ;
```

```
CHAR : char
IDENTIFIER : ch
RIGHT_SQRB : [
INT_CONST : 3
LEFT_SQRB : ]
ASSIGN : =
LEFT_CURL : {
CHAR : 'd'
COMMA : ,
CHAR : 'e'
COMMA : ,
CHAR : '9'
RIGHT_CURL : }
SEMICOLON : ;

STRUCT : struct
LEFT_CURL : {
DOUBLE : double
IDENTIFIER : x
COMMA : ,
IDENTIFIER : y
SEMICOLON : ;
RIGHT_CURL : }
IDENTIFIER : s1
COMMA : ,
IDENTIFIER : s2
COMMA : ,
IDENTIFIER : sm
RIGHT_SQRB : [
INT_CONST : 9
LEFT_SQRB : ]
SEMICOLON : ;

IDENTIFIER : st1
ERROR FLOAT NUM : .
IDENTIFIER : name
ASSIGN : =
STRING : "Bibi"
SEMICOLON : ;
```

```
/* Math Operators */
c = b * b + 3 * b;
p = q % w;
z = z - 1;
x /= y;
y = --x
a = b + c++;

y = x ? a: b;
y = (a>b) ? a: b;
```

```
IT'S A COMMENT : /* Math Operators */

IDENTIFIER : c
ASSIGN : =
IDENTIFIER : b
STAR : *
IDENTIFIER : b
PLUS : +
INT_CONST : 3
STAR : *
IDENTIFIER : b
SEMICOLON : ;

IDENTIFIER : p
ASSIGN : =
IDENTIFIER : q
PERCENT : %
IDENTIFIER : w
SEMICOLON : ;

IDENTIFIER : z
ASSIGN : =
IDENTIFIER : z
MINUS : -
INT_CONST : 1
SEMICOLON : ;

IDENTIFIER : x
ASSIGNMENT OPERATOR : /=
IDENTIFIER : y
SEMICOLON : ;

IDENTIFIER : y
ASSIGN : =
DECREMENT : --
IDENTIFIER : x
```

```
IDENTIFIER : a
ASSIGN : =
IDENTIFIER : b
PLUS : +
IDENTIFIER : c
INCREMENT : ++
SEMICOLON : ;


IDENTIFIER : y
ASSIGN : =
IDENTIFIER : x
QSIGN : ?
IDENTIFIER : a
DPOINT : :
IDENTIFIER : b
SEMICOLON : ;

IDENTIFIER : y
ASSIGN : =
LEFT_PAREN : (
IDENTIFIER : a
RELATIONAL OPERATOR : >
IDENTIFIER : b
RIGHT_PAREN : )
QSIGN : ?
IDENTIFIER : a
DPOINT : :
IDENTIFIER : b
SEMICOLON : ;
```

/* Error Checking */

```
float x = 0..5;
float r = 4.....................5;

char t = '';
char u = '&&'

int 1.x;
```

```
IT'S A COMMENT : /* Error Checking */


FLOAT : float
IDENTIFIER : x
ASSIGN : =
ERROR FLOAT NUM : 0..5
SEMICOLON : ;

FLOAT : float
IDENTIFIER : r
ASSIGN : =
ERROR FLOAT NUM : 4.....................5
SEMICOLON : ;


CHAR : char
IDENTIFIER : t
ASSIGN : =
ERROR CHAR : ''
SEMICOLON : ;

CHAR : char
IDENTIFIER : u
ASSIGN : =
ERROR CHAR : '&&'


INT : int
ERROR ID : 1.x
SEMICOLON : ;
```