```
%{
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

char** identifierTable;
int identifierTableLength = 20;
int identifierPosition = 0;
char** boolTable;
int boolTableLength = 11;
int boolPosition = 0;
char** charTable;
int charTableLength = 20;
int charPosition = 0;
char** stringTable;
int stringTableLength = 20;
int stringPosition = 0;
int* intTable;
int intTableLength = 20;
int intPosition = 0;
struct pifEntry{
    int type; //0-ident, 1-bool, 2-char, 3-string, 4-int, 5-token, 6-separator, 7-operator
    char* token;
    int pos;
};
struct pifEntry* pif;
int pifLength = 20;
int pifPosition = 0;

int lines = 0;

void initTables(){
    //Intialize all the tables
    identifierTable = malloc(identifierTableLength * sizeof(char*));
    stringTable = malloc(stringTableLength * sizeof(char*));
    boolTable = malloc(boolTableLength * sizeof(char*));
    charTable = malloc(charTableLength * sizeof(char*));
    intTable = malloc(intTableLength * sizeof(int));
    pif = malloc(pifLength * sizeof(struct pifEntry));
}

void addIdentifier(char* var){
    //Check if the table needs to be resized
    if (identifierTableLength == identifierPosition){
        char** newIdentifierTable = malloc(identifierTableLength * 2 * sizeof(char*));
        for (int i=0; i<identifierTableLength; i++)
            newIdentifierTable[i] = identifierTable[i];
        free(identifierTable);
        identifierTableLength *= 2;
        identifierTable = newIdentifierTable;
    }
    //Add the variable to the table and increment the position
    identifierTable[identifierPosition] = var;
    identifierPosition++;
}
```

```
void addString(char* var){
    //Check if the table needs to be resized
    if (stringTableLength == stringPosition){
        char** newStringTable = malloc(stringTableLength * 2 * sizeof(char*));
        for (int i=0; i<stringTableLength; i++)
            newStringTable[i] = stringTable[i];
        free(stringTable);
        stringTableLength *= 2;
        stringTable = newStringTable;
    }
    //Add the string to the table and increment the position
    stringTable[stringPosition] = var;
    stringPosition++;
}

void addBool(char* var){
    //Check if the table needs to be resized
    if (boolTableLength == boolPosition){
        char** newBoolTable = malloc(boolTableLength * 2 * sizeof(char*));
        for (int i=0; i<boolTableLength; i++)
            newBoolTable[i] = boolTable[i];
        free(boolTable);
        boolTableLength *= 2;
        boolTable = newBoolTable;
    }
    //Add the variable to the table and increment the position
    boolTable[boolPosition] = var;
    boolPosition++;
}

void addChar(char* var){
    //Check if the table needs to be resized
    if (charTableLength == charPosition){
        char** newCharTable = malloc(charTableLength * 2 * sizeof(char*));
        for (int i=0; i<charTableLength; i++)
            newCharTable[i] = charTable[i];
        free(charTable);
        charTableLength *= 2;
        charTable = newCharTable;
    }
    //Add the variable to the table and increment the position
    charTable[charPosition] = var;
    charPosition++;
}

void addInt(int var){
    //Check if the table needs to be resized
    if (intTableLength == intPosition){
        int* newIntTable = malloc(intTableLength * 2 * sizeof(int));
        for (int i=0; i<intTableLength; i++)
            newIntTable[i] = intTable[i];
        free(intTable);
        intTableLength *= 2;
        intTable = newIntTable;
    }
    //Add the variable to the table and increment the position
    intTable[intPosition] = var;
    intPosition++;
}

void addEntry(struct pifEntry entry){
    //Check if the table needs to be resized
    if (pifLength == pifPosition){
        struct pifEntry* newPif = malloc(pifLength * 2 * sizeof(struct pifEntry));
        for (int i=0; i<pifLength; i++)
            newPif[i] = pif[i];
        free(pif);
        pifLength *= 2;
        pif = newPif;
    }
    //Add the variable to the table and increment the position
    pif[pifPosition] = entry;
    pifPosition++;
}

int getIdentifierIndex(char* var){
    //Check if var is in the table
    for (int i=0; i<identifierPosition; i++)
        if (strcmp(var, identifierTable[i]) == 0)
            return i;
    //Add var to table
    addIdentifier(var);
    return identifierPosition - 1;
}

int getStringIndex(char* var){
    //Check if var is in the table
    for (int i=0; i<stringPosition; i++)
        if (strcmp(var, stringTable[i]) == 0)
            return i;
    //Add var to table
    addString(var);
    return stringPosition - 1;
}

int getBoolIndex(char* var){
    //Check if var is in the table
    for (int i=0; i<boolPosition; i++)
        if (strcmp(var, boolTable[i]) == 0)
            return i;
    //Add var to table
    addBool(var);
    return boolPosition - 1;
}

int getCharIndex(char* var){
    //Check if var is in the table
    for (int i=0; i<charPosition; i++)
        if (strcmp(var, charTable[i]) == 0)
```

```
            return i;
    //Add var to table
    addChar(var);
    return charPosition - 1;
}

int getIntIndex(char* var){
    //Check if var is in the table
    int number = atoi(var);
    for (int i=0; i<intPosition; i++)
        if (number == intTable[i])
            return i;
    //Add var to table
    addInt(number);
    return intPosition - 1;
}

struct pifEntry getEntry(int type, char* token, int position){
    struct pifEntry entry;
    entry.type = type;
    entry.token = token;
    entry.pos = position;
    return entry;
}

char* copyString(char* string){
    //Create new string so the string from yytext can be saved
    int length = (int)strlen(string);
    char* returnString = malloc((length + 1) * sizeof(char));
    for (int i=0; i<=length; i++)
        returnString[i] = string[i];
    return returnString;
}

%}

%option noyywrap

LETTER [a-zA-Z_]
DIGIT [0-9]
NONZERODIGIT [1-9]
IDENTIFIER {LETTER}({LETTER}|{DIGIT})*
BOOLCONST true|false
CHARCONST \'({LETTER}|{DIGIT})\'
INTCONST [+-]?{NONZERODIGIT}{DIGIT}*|0
STRINGCONST \"({LETTER}|{DIGIT}|" ")*\"

%%

"int"|"bool"|"char"|"string"|"if"|"else"|"while"|"print"|"readInt"|"readString"|"array"|"set"|"get" {char* token = co
pyString(yytext); addEntry(getEntry(5, token, -1)); printf("%s - reserved word\n", yytext);}
"{"|"}"|"("|")"|";"|"\""|","|"\'" {char* token = copyString(yytext); addEntry(getEntry(6, token, -1)); printf("%s - se
parator\n", yytext);}
"+"|"-"|"*"|"/"|"%"|"=="|">="|">"|"<="|"<"|"="|"&&"|"||"|"!" {char* token = copyString(yytext); addEntry(getEntry
(7, token, -1)); printf("%s - operator\n", yytext);}
```

```
{BOOLCONST} {char* var = copyString(yytext); addEntry(getEntry(1, NULL, getBoolIndex(var))); printf("%
s - bool\n", yytext);}
{CHARCONST} {char* var = copyString(yytext); addEntry(getEntry(2, NULL, getCharIndex(var))); printf("
%s - char\n", yytext);}
{STRINGCONST} {char* var = copyString(yytext); addEntry(getEntry(3, NULL, getStringIndex(var))); printf
("%s - string\n", yytext);}
{INTCONST} {char* var = copyString(yytext); addEntry(getEntry(4, NULL, getIntIndex(var))); printf("%s - i
nt\n", yytext);}
{IDENTIFIER} {char* var = copyString(yytext); addEntry(getEntry(0, NULL, getIdentifierIndex(var))); printf(
"%s - identifier\n", yytext);}

[ \t]+ {}
[\n]+ {lines++;}

. {printf("Error at token %s at line %d\n", yytext, lines); exit(1);}

%%

int main(int argc, char **argv )
{
    //Check for file
    if (argc > 1)
     yyin = fopen(argv[1], "r");
    else exit(1);

    //Initialize the tables and start scanning
    initTables();
    yylex();

    //Print the tables
    printf("\nInteger Table:\n");
    for (int i=0; i<intPosition; i++)
     printf("%d\n", intTable[i]);
    printf("\n");

    printf("Bool Table:\n");
    for (int i=0; i<boolPosition; i++)
     printf("%s\n", boolTable[i]);
    printf("\n");

    printf("Char Table:\n");
    for (int i=0; i<charPosition; i++)
     printf("%s\n", charTable[i]);
    printf("\n");

    printf("String Table:\n");
    for (int i=0; i<stringPosition; i++)
     printf("%s\n", stringTable[i]);
    printf("\n");

    printf("Identifier Table:\n");
    for (int i=0; i<identifierPosition; i++)
     printf("%s\n", identifierTable[i]);
    printf("\n");
    printf("Pif:\n");
    for (int i=0; i<pifPosition; i++)
     printf("%d - %s - %d\n", pif[i].type, pif[i].token, pif[i].pos);
    printf("\n");

    return 0;
}
```

----

p1.txt

```
int a = 1, b = 2, c = 3;
int min = a;

if (min > a)
{
    min = a;
}
if (min > b)
{
    min = b;
}
print(min);
```

----

out

```
int - reserved word
a - identifier
= - operator
1 - int
, - separator
b - identifier
= - operator
2 - int
, - separator
c - identifier
= - operator
3 - int
; - separator
int - reserved word
min - identifier
= - operator
a - identifier
; - separator
if - reserved word
( - separator
min - identifier
> - operator
a - identifier
) - separator
{ - separator
min - identifier
= - operator
a - identifier
; - separator
} - separator
if - reserved word
( - separator
min - identifier
> - operator
b - identifier
) - separator
{ - separator
min - identifier
= - operator
b - identifier
; - separator
} - separator
print - reserved word
( - separator
min - identifier
) - separator
; - separator

Integer Table:
1
2
3

Bool Table:

Char Table:

String Table:

Identifier Table:
a
b
c
min

Pif:
5 - int - -1
0 - (null) - 0
7 - = - -1
4 - (null) - 0
6 - , - -1
0 - (null) - 1
7 - = - -1
4 - (null) - 1
6 - , - -1
0 - (null) - 2
7 - = - -1
4 - (null) - 2
6 - ; - -1
5 - int - -1
0 - (null) - 3
```

```
7 - = - -1
0 - (null) - 0
6 - ; - -1
5 - if - -1
6 - ( - -1
0 - (null) - 3
7 - > - -1
0 - (null) - 0
6 - ) - -1
6 - { - -1
0 - (null) - 3
7 - = - -1
0 - (null) - 0
6 - ; - -1
6 - } - -1
5 - if - -1
6 - ( - -1
0 - (null) - 3
7 - > - -1
0 - (null) - 1
6 - ) - -1
6 - { - -1
0 - (null) - 3
7 - = - -1
0 - (null) - 1
6 - ; - -1
6 - } - -1
5 - print - -1
6 - ( - -1
0 - (null) - 3
6 - ) - -1
6 - ; - -1
```

----
p2.txt

```
int x = 12;
int i = 2;
bool prime = true;
while (i*i <= x && prime)
{
    if (x % i == 0)
    {
        prime = false;
    }
    i = i + 1;
}
print(prime);
```

----
out

```
int - reserved word
x - identifier
```

```
= - operator
12 - int
; - separator
int - reserved word
i - identifier
= - operator
2 - int
; - separator
bool - reserved word
prime - identifier
= - operator
true - bool
; - separator
while - reserved word
( - separator
i - identifier
* - operator
i - identifier
<= - operator
x - identifier
&& - operator
prime - identifier
) - separator
{ - separator
if - reserved word
( - separator
x - identifier
% - operator
i - identifier
== - operator
0 - int
) - separator
{ - separator
prime - identifier
= - operator
false - bool
; - separator
} - separator
i - identifier
= - operator
i - identifier
+ - operator
1 - int
; - separator
} - separator
print - reserved word
( - separator
prime - identifier
) - separator
; - separator

Integer Table:
12
2
0
```

```
1

Bool Table:
true
false

Char Table:

String Table:

Identifier Table:
x
i
prime

Pif:
5 - int - -1
0 - (null) - 0
7 - = - -1
4 - (null) - 0
6 - ; - -1
5 - int - -1
0 - (null) - 1
7 - = - -1
4 - (null) - 1
6 - ; - -1
5 - bool - -1
0 - (null) - 2
7 - = - -1
1 - (null) - 0
6 - ; - -1
5 - while - -1
6 - ( - -1
0 - (null) - 1
7 - * - -1
0 - (null) - 1
7 - <= - -1
0 - (null) - 0
7 - && - -1
0 - (null) - 2
6 - ) - -1
6 - { - -1
5 - if - -1
6 - ( - -1
0 - (null) - 0
7 - % - -1
0 - (null) - 1
7 - == - -1
4 - (null) - 2
6 - ) - -1
6 - { - -1
0 - (null) - 2
7 - = - -1
1 - (null) - 1
6 - ; - -1
6 - } - -1
0 - (null) - 1
7 - = - -1
0 - (null) - 1
7 - + - -1
4 - (null) - 3
6 - ; - -1
6 - } - -1
5 - print - -1
6 - ( - -1
0 - (null) - 2
6 - ) - -1
6 - ; - -1
```

----
p3.txt

```
int n = 5;
int sum = 0;
int i = 0;
int var;
while (i < n)
{
    var = readInt();
    sum = sum + var;
    i = i + 1;
}
```

----
out

```
int - reserved word
n - identifier
= - operator
5 - int
; - separator
int - reserved word
sum - identifier
= - operator
0 - int
; - separator
int - reserved word
i - identifier
= - operator
0 - int
; - separator
int - reserved word
var - identifier
; - separator
while - reserved word
( - separator
i - identifier
< - operator
n - identifier
) - separator
{ - separator
var - identifier
= - operator
readInt - reserved word
( - separator
) - separator
; - separator
sum - identifier
= - operator
sum - identifier
+ - operator
var - identifier
; - separator
i - identifier
= - operator
i - identifier
+ - operator
1 - int
; - separator
} - separator

Integer Table:
5
0
1

Bool Table:

Char Table:

String Table:

Identifier Table:
n
sum
i
var

Pif:
5 - int - -1
0 - (null) - 0
7 - = - -1
4 - (null) - 0
6 - ; - -1
5 - int - -1
0 - (null) - 1
7 - = - -1
4 - (null) - 1
6 - ; - -1
5 - int - -1
0 - (null) - 2
7 - = - -1
4 - (null) - 1
6 - ; - -1
```

```
5 - int - -1
0 - (null) - 3
6 - ; - -1
5 - while - -1
6 - ( - -1
0 - (null) - 2
7 - < - -1
0 - (null) - 0
6 - ) - -1
6 - { - -1
0 - (null) - 3
7 - = - -1
5 - readInt - -1
6 - ( - -1
6 - ) - -1
6 - ; - -1
0 - (null) - 1
7 - = - -1
0 - (null) - 1
7 - + - -1
0 - (null) - 3
6 - ; - -1
0 - (null) - 2
7 - = - -1
0 - (null) - 2
7 - + - -1
4 - (null) - 2
6 - ; - -1
6 - } - -1
```

----

perr.txt

```
int 5n = 5;
int sum = 015;
int i = 0;
int var;
while (i < n)
{
    var = readInt();
    sum = sum + var;
    i = i + 1;
}
```

----

out

```
int - reserved word

Error at token 5 at line 0
```