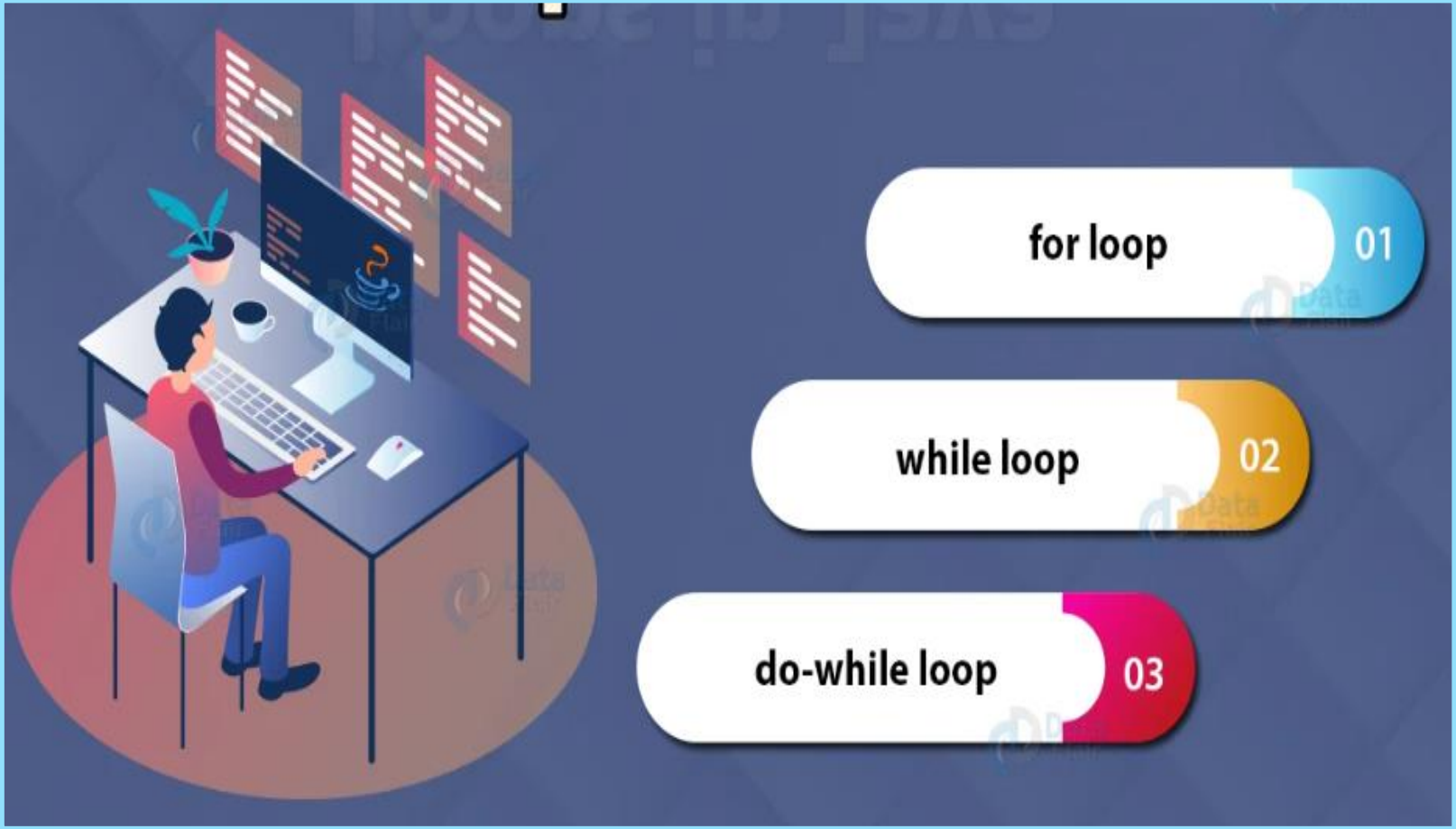
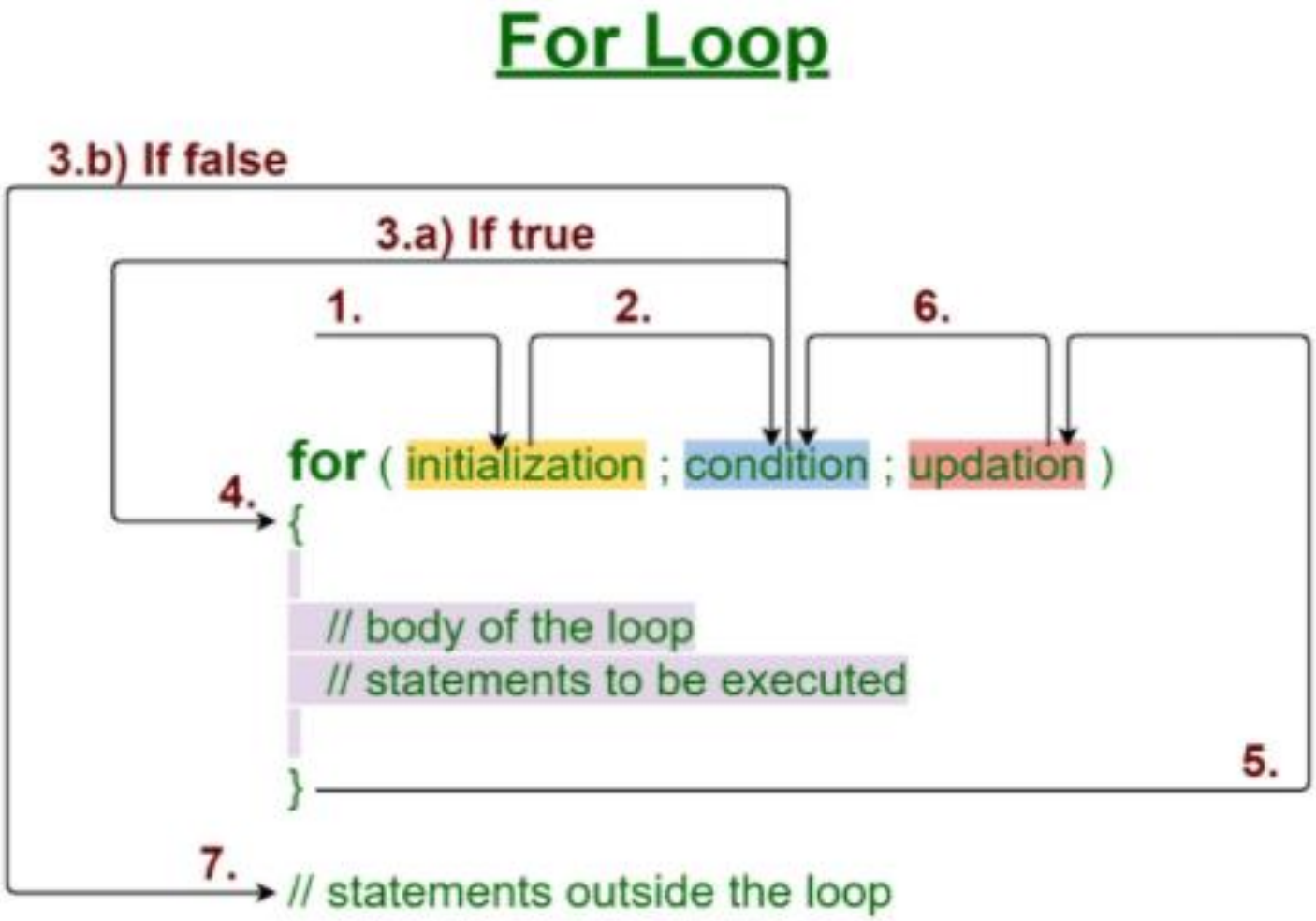
Parsing LOOP statements



Parsing FOR statement



Parsing FOR statement

CORRECT:

```
for(int i=1;i<10;i++)
{
    f15=99;
    m3=v77;
}
```

Parsed FOR statement

UNCORRECT:

```
for(i=1 int ; i++; i<10)
{
    f15==99:
    3m=v77;
}
```

Error parsing FOR statement

Parsing FOR statement

TEXT file1:(**for1.t**)

```
for(int i=100;i<15;i--)
{
    f15=99;
    m3=v77;
}
```

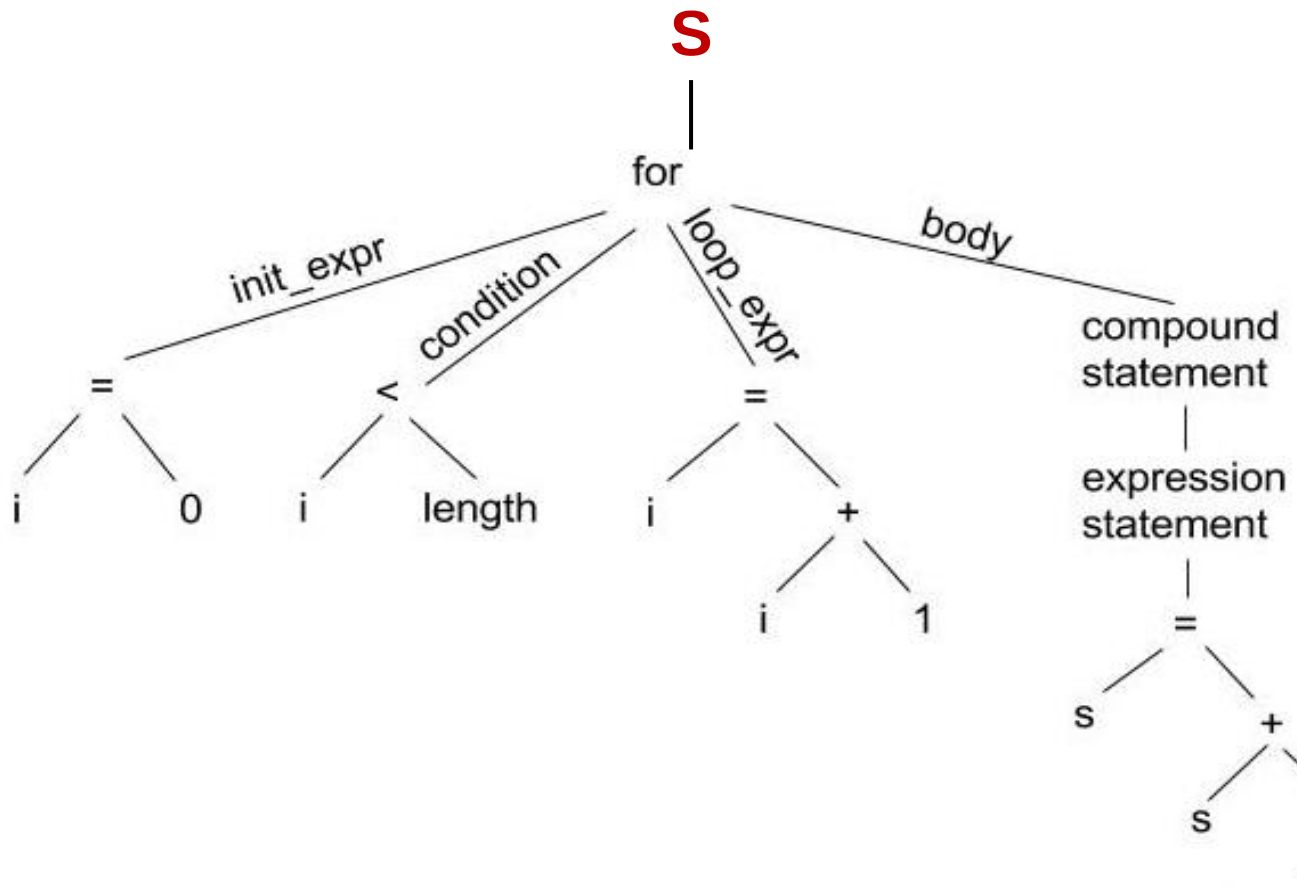
Parsed FOR statement

Parsing FOR statement

LEX file1:(**for1.l**)

```
%%
for  return FOR;
int  return INT;
[a-z]+[a-z0-9]*  return ID;
[0-9]+    return NUM;
[ \t] printf(" ");
.   return yytext[0];
.|\n  ;
%%
```

Parsing FOR statement



YACC recognize the language construct or, in other words, **how the start symbol of your grammar** (**S**) **derives a certain string in the programming language**.

Parsing FOR statement

YACC file1:(**for1.y**)

```
%{
#include<stdio.h>
int yylex();
int yyerror();
%}
%token FOR INT ID NUM
%%
S: ST { printf("Parsed FOR statement\n"); }
ST: FOR '(' INT  ASS1  COND ';' ASS2 ')' '{' BLOCK '}';
ASS1: ID '=' NUM ';' | ID '=‘ ID ';';
ASS2: ID '+' '+' | ID '-''-' ;
COND: ID '<' ID | ID '>' ID | ID '<' NUM | ID '>' NUM;
BLOCK: BLOCK  ASS1 | ASS1;
%%
```

Parsing FOR statement

YACC file1:(**for1.y**)

```
#include "lex.yy.c"
int main()
{
   return yyparse();
}
int yyerror()
{
  printf("Error parsing FOR statement \n");
  return 0;
}
```

Parsing FOR statement

nano for1.t — בניית קובץ טקסט לבדיקת תחבירית

nano for1.l — בניית קובץ LEX

nano for1.y — בניית קובץ YACC

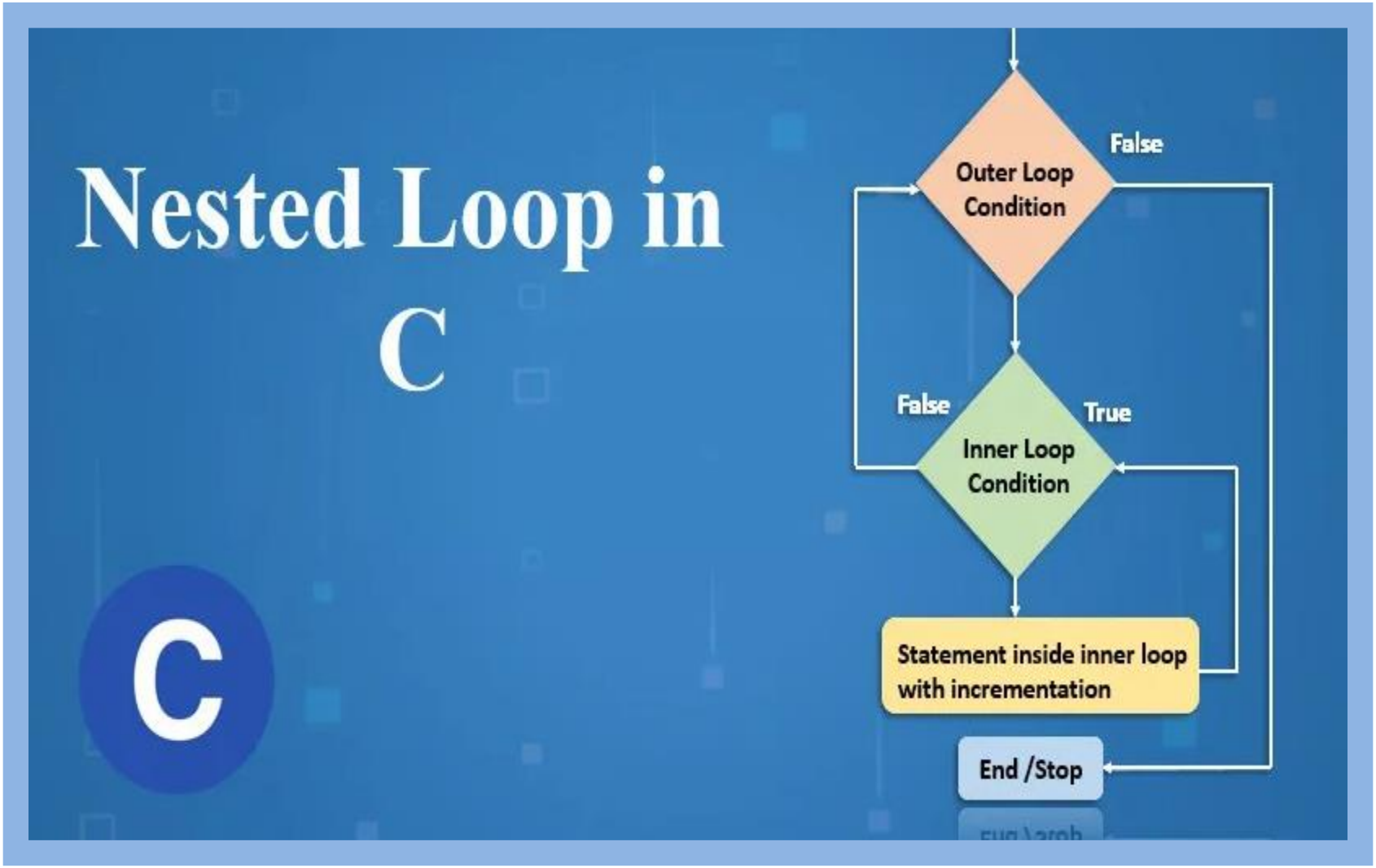
lex for1.l — קומפילציה LEX

yacc for1.y — קומפילציה YACC

cc -o for1 y.tab.c –ll –Ly — קומפילציה C

./for1<for1.t — הרצת הקובץ for1

Parsing FOR statement



Parsing FOR statement

TEXT file2:(**for2.t**)

```
for(int i3=10;i3<100;i++)
{
    f15=99;
    m3=v77;
    for(int j1= 50;j1>0;j1--)
    {
        s33=15;
        m5=v77;
    }
    k15=a99;
}
```

Nested FOR statements

Parsing FOR statement

LEX file2:(**for2.l**)

```
%%
for  return FOR;
int  return INT;
[a-z]+[a-z0-9]*  return ID;
[0-9]+    return NUM;
[ \t] printf(" ");
.   return yytext[0];
.|\n  ;
%%
```

Parsing FOR statement

YACC file2:(**for2.y**)

```
%{
#include<stdio.h>
int yylex();
int yyerror();
%}
%token FOR INT ID NUM
%%
S: ST { printf("Parsed FOR statement\n"); }
ST: FOR '(' INT  ASS1  COND ';' ASS2 ')' '{' BLOCK '}';
ASS1: ID '=' NUM ';' | ID '=' ID ';';
ASS2: ID '+' '+' | ID '-"-' ;
COND: ID '<' ID | ID '>' ID | ID '<' NUM | ID '>' NUM;
BLOCK: BLOCK ASS1| ASS1| BLOCK ST | ST;
%%
```

Parsing FOR statement

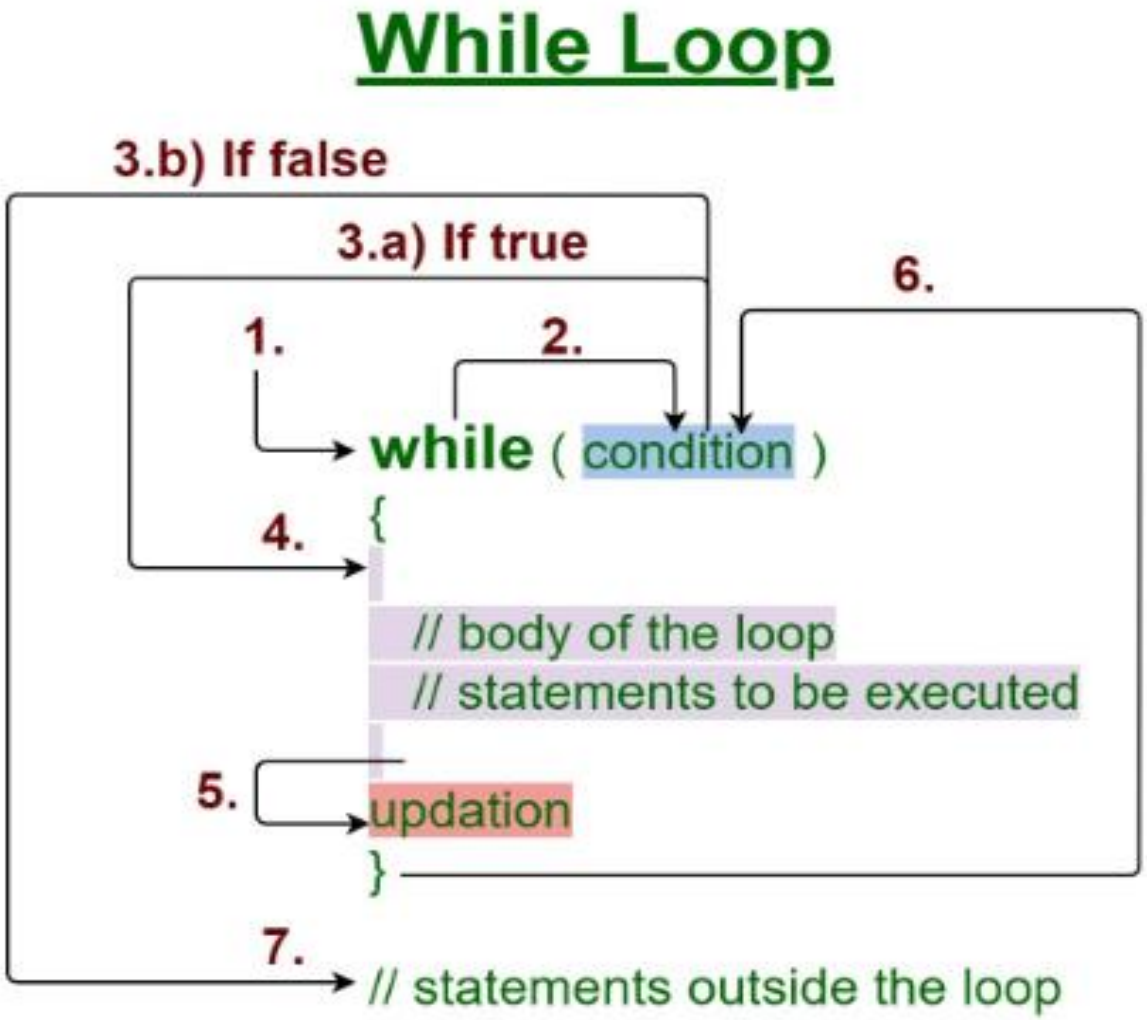
YACC file2:(**for2.y**)

```
#include "lex.yy.c"
int main()
{
   return yyparse();
}
int yyerror()
{
  printf(" Error parsing FOR statements \n ");
  return 0;
}
```

Parsing FOR statement

nano for2.t	בניית קובץ טקסט
nano for2.l	בניית קובץ LEX
nano for2.y	בניית קובץ YACC
lex for2.l	קומפילציה LEX
yacc for2.y	קומפילציה YACC
cc -o for2 y.tab.c –ll –Ly	קומפילציה C
./for2<for2.t	הרצת הקובץ for2

Parsing WHILE statement



Parsing WHILE statement

CORRECT:

```
while(a1!=b2)
{
    f15=b52;
    h3=88;
}
```

Parsed WHILE statement

UNCORRECT:

```
whyle(a1=! b2)
{
    15f=b52;
    h3=88:
}
```

Error parsing WHILE statement

Parsing WHILE statement

TEXT file (**while.t**)

```
while(a1==b2)
{
    f15=b52;
    s3=88;
    if(x>21)
        e4=77;
    else
        v33=b17;
    d4=g33;
}
```

Parsing WHILE statement

LEX file (**while.l**)

```
%%
while   return WHILE;
if  return IF;
else  return ELSE;
[a-z]+[a-z0-9]*  return ID;
[0-9]+  return NUM;
"=="  return EQ;
[ \t]  printf(" ");
.  return yytext[0];
.|\n  ;
%%
```

Parsing WHILE statement

YACC file (**while.y**)

```
%{
#include<stdio.h>
int yyerror();
int yylex();
%}
%token WHILE  ID  NUM  EQ  IF  ELSE
%%
S: EX { printf("INPUT ACCEPTED\n"); }
EX: WHILE  '('  COND  ')'  '{'  BLOCK  '}';
COND: ID EQ NUM|ID'<'NUM|ID'>'NUM|ID EQ ID|ID'>'ID|ID'<'ID;
BLOCK: BLOCK  ST | ST;
ST: IF_ST | ASS_ST;
IF_ST: IF '(' COND ')'  ASS_ST ELSE  ASS_ST;
ASS_ST: ID '=' ID ';' | ID '=' NUM ';' ;
%%
```

Parsing WHILE statement

```
#include "lex.yy.c"
int main()
{
  return yyparse();
}
int yyerror()
{
   printf("ERROR\n");
   return 0;
}
```

YACC file (**while.y**)

Parsing WHILE statement

nano while.t	בניית קובץ טקסט
nano while.l	בניית קובץ LEX
nano while.y	בניית קובץ YACC
lex while.l	קומפילציה LEX
yacc while.y	קומפילציה YACC
cc -o while y.tab.c –ll –Ly	קומפילציה C
./while<while.t	הרצת הקובץ while