

Name:- Chakshu Saraswat
Semester:- 5th
Section:- C
Registration Number:- 180905482
Roll No.:- 57

CD LAB 9

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include "lex_analyzer.h"

void program();
void declarations();
void datatype();
void idlist();
void idlistprime();
void assignstat();
void statement_list();
void statement();
void expn();
void eprime();
void simpleexp();
void seprime();
void term();
void tprime();
void factor();
void decision_stat();
void dprime();
void looping_stat();
void relop();
void addop();
void mulop();
void printerror(struct token*);

void printerror(struct token* tkn)
{
    printf("error at row: %d, col: %d for lexeme \" %s \"\n", tkn->row, tkn->col,
tkn->lexeme);
    printf("-----ERROR!-----\n");
    exit(0);
}

// all declarations
struct token tkn;
```

```

FILE *f1;
char *rel[]={"==","!=","<=",">=",">","<"};
char *add[]={"+","-"};
char *mul[]={"*","/","%"};

//check for comparison operators
int isrel(char *w)
{
    int i;
    for(i=0;i<sizeof(rel)/sizeof(char*);i++)
    {
        if(strcmp(w,rel[i])==0)
        {
            return 1;
        }
    }
    return 0;
}

//check for +, - operators
int isadd(char *w)
{
    int i;
    for(i=0;i<sizeof(add)/sizeof(char*);i++)
    {
        if(strcmp(w,add[i])==0)
        {
            return 1;
        }
    }
    return 0;
}

//check for *, /, % operators
int ismul(char *w)
{
    int i;
    for(i=0;i<sizeof(mul)/sizeof(char*);i++)
    {
        if(strcmp(w,mul[i])==0)
        {
            return 1;
        }
    }
    return 0;
}

void program()
{
    if(strcmp(tkn.lexeme,"main")==0)
    {

```

```

tkn=getNextToken(f1);
if(strcmp(tkn.lexeme,"")==0)
{
    tkn=getNextToken(f1);
    if(strcmp(tkn.lexeme,"")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"{")==0)
        {
            tkn=getNextToken(f1);
            declarations();
            statement_list();
            if(strcmp(tkn.lexeme,"}")==0)
            {
                return;
            }
            else if(strcmp(tkn.lexeme,"for")==0 ||
// loop beginning
{
                looping_stat();
                if(strcmp(tkn.lexeme,"}")==0)
                {
                    return;
                    exit(0);
                }
                else if(strcmp(tkn.lexeme,"for")==0 ||
// loop beginning
{
                    looping_stat();
                }
                else if(strcmp(tkn.lexeme,"if")==0)
                {
                    decision_stat();
                }
                else
                {
                    printf("{} missing at row=%d",
tkn.row,tkn.col);
                    exit(1);
                }
            }
            else if(strcmp(tkn.lexeme,"if")==0) // conditional
            statement beginning
            {
                decision_stat();
                if(strcmp(tkn.lexeme,"}")==0)
                {
                    return;
                }
                else if(strcmp(tkn.lexeme,"for")==0 ||
// loop beginning
{
                    looping_stat();
                }
                else if(strcmp(tkn.lexeme,"if")==0)
                {
                    decision_stat();
                }
                else
                {
                    printf("{} missing at row=%d",
tkn.row,tkn.col);
                    exit(1);
                }
            }
        }
    }
}

```



```

        printerror(&tkn);
    }
}

//check for the correct datatype
void datatype()
{
    if(strcmp(tkn.lexeme,"int")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
    else if(strcmp(tkn.lexeme,"char")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
    else
    {
        printerror(&tkn);
    }
}

void idlist()
{
    if(strcmp(tkn.type,"identifier")==0)
    {
        tkn=getNextToken(f1);
        idlistprime();
    }
    else
    {
        printerror(&tkn);
    }
}

void idlistprime()
{
    if(strcmp(tkn.lexeme,",")==0)
    {
        tkn=getNextToken(f1);
        idlist();
    }
    if(strcmp(tkn.lexeme,"[")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.type,"number")==0)
        {
            tkn=getNextToken(f1);
            if(strcmp(tkn.lexeme,"]")==0)
            {
                tkn=getNextToken(f1);
                if(strcmp(tkn.lexeme,",")==0)

```

```

        {
            tkn=getNextToken(f1);
            idlist();
        }
        else
        {
            return;
        }
    }
    else
    {
        printerror(&tkn);
    }
}
}
else
{
    return;
}
}
void statement_list()
{
    if(strcmp(tkn.type,"identifier")!=0)
    {
        return;
    }
    statement();
    statement_list();
}
void statement()
{
    if(strcmp(tkn.type,"identifier")==0)
    {
        assignstat();
        if(strcmp(tkn.lexeme,";")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        else
        {
            printerror(&tkn);
        }
    }
    if(strcmp(tkn.lexeme,"if")==0)
    {
        decision_stat();
    }
    if(strcmp(tkn.lexeme,"while")==0 || strcmp(tkn.lexeme,"for")==0)
    {
        looping_stat();
    }
}

```

```

}
void assignstat()
{
    if(strcmp(tkn.type,"identifier")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"")==0)
        {
            tkn=getNextToken(f1);
            expn();
        }
        else
        {
            printerror(&tkn);
        }
    }
    else
    {
        printerror(&tkn);
    }
}
void expn()
{
    simpleexp();
    eprime();
}
void eprime()
{
    if(isrel(tkn.lexeme)==0)
    {
        return;
    }
    relop();
    simpleexp();
}
void simpleexp()
{
    term();
    seprime();
}
void seprime()
{
    if(isadd(tkn.lexeme)==0)
    {
        return;
    }
    addop();
    term();
    seprime();
}
void term()
{

```

```

        factor();
        tprime();
    }
void tprime()
{
    if(ismul(tkn.lexeme)==0)
    {
        return;
    }
    mulop();
    factor();
    tprime();
}
void factor()
{
    if(strcmp(tkn.type,"identifier")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
    else if(strcmp(tkn.type,"number")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
}
void decision_stat()
{
    if(strcmp(tkn.lexeme,"if")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"(")==0)
        {
            tkn=getNextToken(f1);
            expn();
            if(strcmp(tkn.lexeme,"")==0)
            {
                tkn=getNextToken(f1);
                if(strcmp(tkn.lexeme,"{")==0)
                {
                    tkn=getNextToken(f1);
                    statement_list();
                    if(strcmp(tkn.lexeme,"}")==0)
                    {
                        tkn=getNextToken(f1);
                        dprime();
                    }
                    else
                    {
                        printerror(&tkn);
                    }
                }
            }
        }
    }
}

```



```

        else
        {
            printerror(&tkn);
        }
    }
    else
    {
        printerror(&tkn);
    }
}
else
{
    printerror(&tkn);
}
}
}
void dprime()
{
    if(strcmp(tkn.lexeme,"else")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"{")==0)
        {
            tkn=getNextToken(f1);
            statement_list();
            if(strcmp(tkn.lexeme,"}")==0)
            {
                tkn=getNextToken(f1);
                return;
            }
            else
            {
                printerror(&tkn);
            }
        }
        else
        {
            printerror(&tkn);
        }
    }
    else
    {
        return;
    }
}
void looping_stat()
{
    if(strcmp(tkn.lexeme,"while")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"(")==0)
        {

```

```

    tkn=getNextToken(f1);
    expn();
    if(strcmp(tkn.lexeme,"")==0)
    {
        tkn=getNextToken(f1);
        if(strcmp(tkn.lexeme,"{")==0)
        {
            tkn=getNextToken(f1);
            statement_list();
            if(strcmp(tkn.lexeme,"}")==0)
            {
                tkn=getNextToken(f1);
                return;
            }
            else
            {
                printerror(&tkn);
            }
        }
        else
        {
            printerror(&tkn);
        }
    }
    else
    {
        printerror(&tkn);
    }
}
else if(strcmp(tkn.lexeme,"for")==0)
{
    tkn=getNextToken(f1);
    if(strcmp(tkn.lexeme,"(")==0)
    {
        tkn=getNextToken(f1);
        assignstat();
        if(strcmp(tkn.lexeme,";")==0)
        {
            tkn=getNextToken(f1);
            expn();
            if(strcmp(tkn.lexeme,";")==0)
            {
                tkn=getNextToken(f1);
                assignstat();
                if(strcmp(tkn.lexeme,"")==0)
                {
                    tkn=getNextToken(f1);

```

```

        if(strcmp(tkn.lexeme,"{")==0)
        {
            tkn=getNextToken(f1);
            statement_list();
            if(strcmp(tkn.lexeme,"}")==0)
            {
                tkn=getNextToken(f1);
                return;
            }
            else
            {
                printerror(&tkn);
            }
        }
        else
        {
            printerror(&tkn);
        }
    }
    else
    {
        printerror(&tkn);
    }
}
else
{
    printerror(&tkn);
}
}
else
{
    printerror(&tkn);
}
}
}

void relop()
{
    if(strcmp(tkn.lexeme,"")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
    if(strcmp(tkn.lexeme,"!=")==0)
    {
        tkn=getNextToken(f1);
        return;
    }
}

```

```

        if(strcmp(tkn.lexeme,"<")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,">")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,"<")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,">")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
    }
    void addop()
    {
        if(strcmp(tkn.lexeme,"+")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,"-")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
    }
    void mulop()
    {
        if(strcmp(tkn.lexeme,"*")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,"/")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
        if(strcmp(tkn.lexeme,"%")==0)
        {
            tkn=getNextToken(f1);
            return;
        }
    }
}

```

```

int main()
{
    FILE *fa, *fb;
    int ca, cb;
    fa = fopen("inp.c", "r");
    if (fa == NULL){
        printf("Cannot open file \n");
        exit(0);
    }

    fb = fopen("out.c", "w+");
    ca = getc(fa);
    while (ca != EOF){
        if(ca==' ')
        {
            putc(ca,fb);
            while(ca==' ')
                ca = getc(fa);
        }
        if (ca=='/')
        {
            cb = getc(fa);
            if (cb == '/')
            {
                while(ca != '\n')
                    ca = getc(fa);
            }
            else if (cb == '*')
            {
                do
                {
                    while(ca != '*')
                        ca = getc(fa);
                    ca = getc(fa);
                } while (ca != '/');
            }
            else{
                putc(ca,fb);
                putc(cb,fb);
            }
        }
        else putc(ca,fb);
        ca = getc(fa);
    }
    fclose(fa);
    fclose(fb);
    fa = fopen("out.c", "r");
    if(fa == NULL){
        printf("Cannot open file");
        return 0;
    }
}

```

```

fb = fopen("temp.c", "w+");
ca = getc(fa);
while (ca != EOF)
{
    if(ca=="")
    {
        putc(ca,fb);
        ca=getc(fa);
        while(ca!="")
        {
            putc(ca,fb);
            ca=getc(fa);
        }
    }
    else if(ca=='#')
    {
        while(ca!='\n')
        {
            ca=getc(fa);
        }
        ca=getc(fa);
    }
    putc(ca,fb);
    ca = getc(fa);
}

fclose(fa);
fclose(fb);

fa = fopen("temp.c", "r");
fb = fopen("out.c", "w");
ca = getc(fa);
while(ca != EOF){
    putc(ca, fb);
    ca = getc(fa);
}
fclose(fa);
fclose(fb);
remove("temp.c");
f1=fopen("out.c","r");
if(f1==NULL)
{
    printf("Error! File cannot be opened!\n");
    return 0;
}

while((tkn=getNextToken(f1)).row!=-1)
{
    if(strcmp(tkn.lexeme,"main")==0)
    {

```

```

        program();
        break;
    }
}

printf("Compiled sucessfully\n");
fclose(f1);
}

```

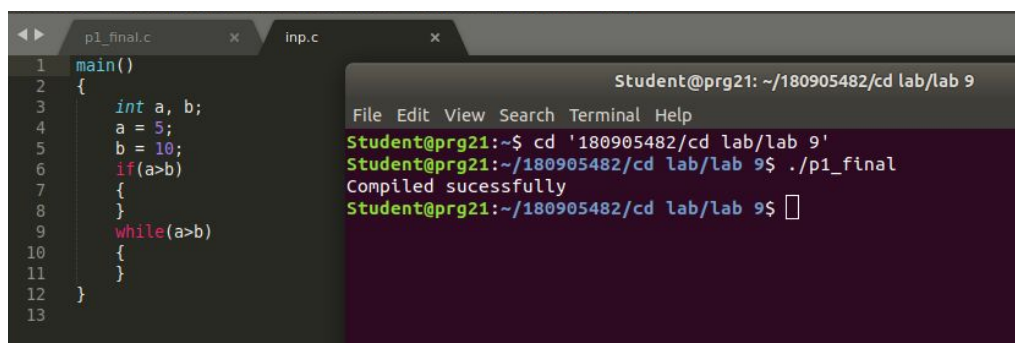
OUTPUT:-

Case 1:-

```

main()
{
    int a, b;
    a = 5;
    b = 10;
    if(a>b)
    {
    }
    while(a>b)
    {
    }
}

```



The screenshot shows a code editor with two tabs: 'p1_final.c' and 'inp.c'. The 'p1_final.c' tab is active, displaying the following C code:

```

1  main()
2  {
3      int a, b;
4      a = 5;
5      b = 10;
6      if(a>b)
7      {
8      }
9      while(a>b)
10     {
11     }
12 }
13

```

To the right of the code editor is a terminal window titled 'Student@prg21: ~/180905482/cd lab/lab 9'. The terminal shows the following commands and output:

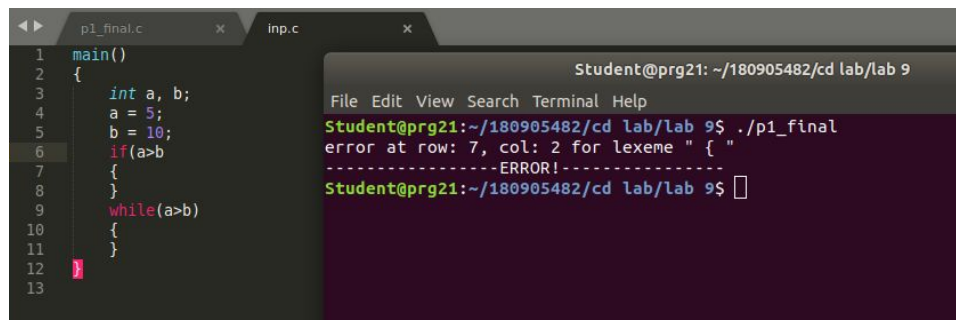
```

Student@prg21:~$ cd '180905482/cd lab/lab 9'
Student@prg21:~/180905482/cd lab/lab 9$ ./p1_final
Compiled sucessfully
Student@prg21:~/180905482/cd lab/lab 9$ 

```

Case 2:-

```
main()
{
    int a, b;
    a = 5;
    b = 10;
    if(a>b
    {
    }
    while(a>b)
    {
    }
}
```



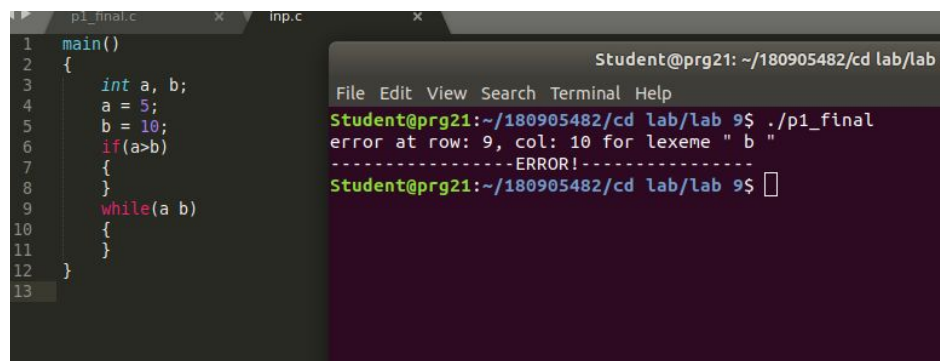
The screenshot shows a code editor with two tabs: 'p1_final.c' and 'inp.c'. The 'p1_final.c' tab is active, displaying the following code:

```
1 main()
2 {
3     int a, b;
4     a = 5;
5     b = 10;
6     if(a>b
7     {
8     }
9     while(a>b)
10    {
11    }
12
13
```

The code is syntactically incorrect due to a missing closing brace for the 'if' statement on line 6. To the right of the code editor is a terminal window with the title 'Student@prg21: ~/180905482/cd lab/lab 9'. The terminal shows the command './p1_final' being executed, which results in the error message: 'error at row: 7, col: 2 for lexeme " { "'. The terminal also displays '-----ERROR!-----' and the prompt 'Student@prg21:~/180905482/cd lab/lab 9\$'.

Case 3:-

```
main()
{
    int a, b;
    a = 5;
    b = 10;
    if(a>b)
    {
    }
    while(a b)
    {
    }
}
```



The screenshot shows a code editor with a file named `p1_final.c` open. The code in the editor is as follows:

```
1 main()
2 {
3     int a, b;
4     a = 5;
5     b = 10;
6     if(a>b)
7     {
8     }
9     while(a b)
10    {
11    }
12 }
13
```

To the right of the code editor is a terminal window titled "Student@prg21: ~/180905482/cd lab/lab". The terminal shows the command `./p1_final` being executed, which results in a syntax error:

```
Student@prg21:~/180905482/cd lab/lab 9$ ./p1_final
error at row: 9, col: 10 for lexeme " b "
-----ERROR!-----
Student@prg21:~/180905482/cd lab/lab 9$
```

The error message indicates that there is a syntax error at row 9, column 10, for the lexeme " b ". This is because the `while` loop condition `a b` is missing an operator between `a` and `b`.