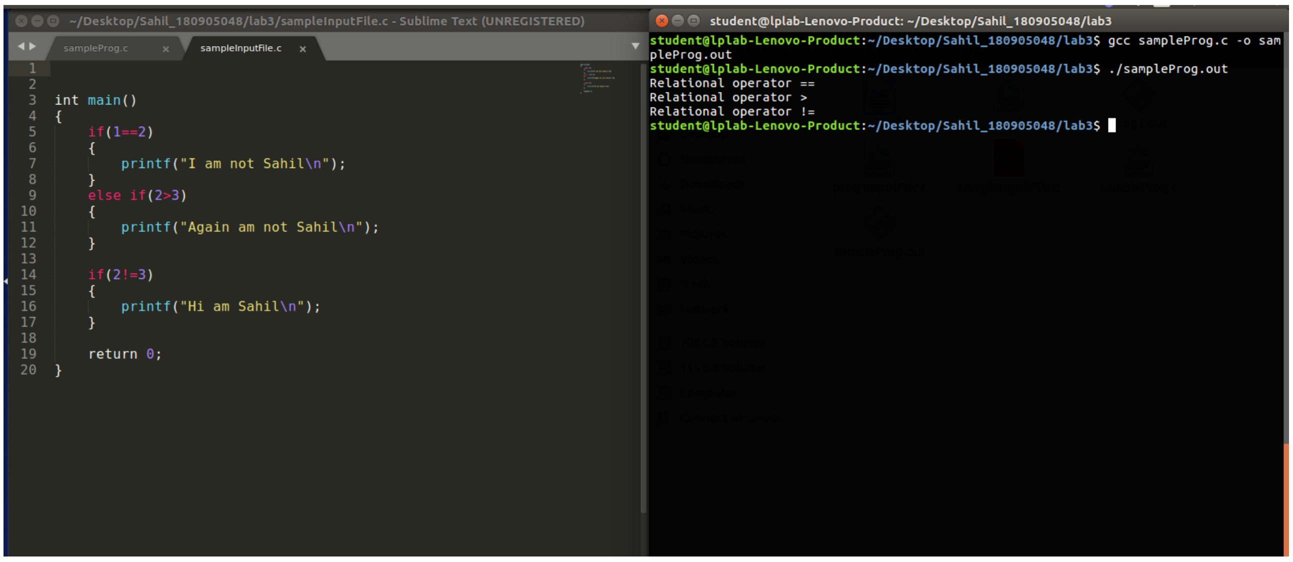
LAB 3

Sahil Saini Salaria
Reg No. 180905048
Roll no. 11C
Lab CD

```
// Solved problem

#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main()
{
	char ca,cb;
	char buff[100];
	int k=0;

	FILE *fa=fopen("sampleInputFile.c","r");
	if(fa==NULL)
	{
		printf("Cannot open the file\n");
		exit(0);
	}

	ca=getc(fa);

	while(ca!=EOF)
	{
		k=0;

		if(ca=='=')
		{
			buff[k++]=ca;
			cb=getc(fa);
			if(cb=='=')
			{
				buff[k++]=cb;
				buff[k++]='\0';
				printf("Relational operator %s\n",buff);
			}
			else
			{
				buff[k++]='\0';
				printf("Assignment operator\n");
			}
		}
		else
		{
```

```
                        if(ca=='<'|| ca=='>' || ca=='!')
                        {
                                buff[k++]=ca;
                                cb=getc(fa);
                                if(cb=='=')
                                {
                                        buff[k++]=cb;
                                }
                                buff[k++]='\0';
                                printf("Relational operator %s\n",buff);
                        }
                        else
                        {
                                buff[k++]='\0';
                        }
                }

                ca=getc(fa);
        }
        return 0;
}
```



~/Desktop/Sahil_180905048/lab3/sampleInputFile.c - Sublime Text (UNREGISTERED)

```

int main()
{
    if(1==2)
    {
        printf("I am not Sahil\n");
    }
    else if(2>3)
    {
        printf("Again am not Sahil\n");
    }

    if(2!=3)
    {
        printf("Hi am Sahil\n");
    }

    return 0;
}
```

student@lplab-Lenovo-Product: ~/Desktop/Sahil_180905048/lab3

```
student@lplab-Lenovo-Product:~/Desktop/Sahil_180905048/lab3$ gcc sampleProg.c -o sam
pleProg.out
student@lplab-Lenovo-Product:~/Desktop/Sahil_180905048/lab3$ ./sampleProg.out
Relational operator ==
Relational operator >
Relational operator !=
student@lplab-Lenovo-Product:~/Desktop/Sahil_180905048/lab3$
```

Q1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define FILEINPUT "sample.c"
struct token
```

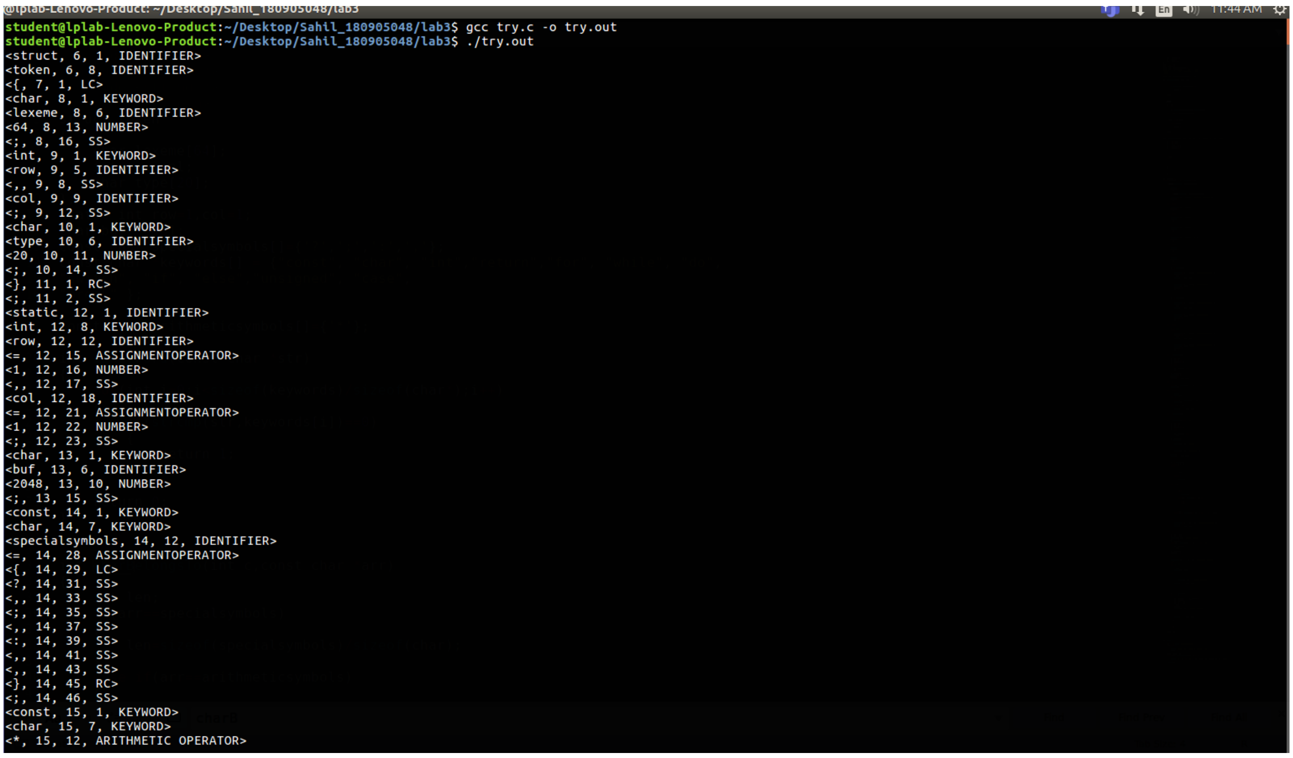
```
{
	char lexeme[64];
	int row,col;
	char type[20];
};
static int row=1,col=1;
char buf[2048];
const char specialsymbols[]={'?',';',':',','};
const char *keywords[] = {"const", "char", "int","return","for", "while", "do",
"switch", "if", "else","unsigned", "case",
"break" };

const char arithmeticsymbols[]={'*'};

int isKeyword(const char *str)
{
	for(int i=0;i<sizeof(keywords)/sizeof(char*);i++)
	{
		if(strcmp(str,keywords[i])==0)
		{
			return 1;
		}
	}
	return 0;
}


int charBelongsTo(int c,const char *arr)
{
	int len;
	if(arr==specialsymbols)
	{
		len=sizeof(specialsymbols)/sizeof(char);
	}
	else if(arr==arithmeticsymbols)
	{
		len=sizeof(arithmeticsymbols)/sizeof(char);
	}
	for(int i=0;i<len;i++)
	{
		if(c==arr[i])
		{
			return 1;
		}
	}
	return 0;
}


void fillToken(struct token *tkn,char c,int row,int col, char *type)
{
	tkn->row=row;
	tkn->col=col;
	strcpy(tkn->type,type);
	tkn->lexeme[0]=c;
	tkn->lexeme[1]='\0';
}

void newLine()
{
	++row;
	col=1;
}

struct token getNextToken(FILE *f1)
{
	int c;
	struct token tkn=
	{
		.row=-1
	};


int gotToken=0;
while(!gotToken && (c=fgetc(f1))!=EOF)
{
	if(charBelongsTo(c,specialsymbols))
	{
		fillToken(&tkn,c,row,col,"SS");
		gotToken=1;
		++col;
	}
	else if(charBelongsTo(c,arithmeticsymbols))
	{
		fillToken(&tkn,c,row,col,"ARITHMETIC OPERATOR");
		gotToken=1;
		++col;
	}
	else if(c=='(')
	{
		fillToken(&tkn,c,row,col,"LB");
		gotToken=1;
		++col;
	}
	else if(c==')')
	{
		fillToken(&tkn,c,row,col,"RB");
		gotToken=1;
		++col;
	}
	else if(c=='{')
	{
		fillToken(&tkn,c,row,col,"LC");
		gotToken=1;
		++col;
	}
	else if(c=='[')
	{
		fillToken(&tkn,c,row,col,"LSB");
		gotToken=1;
		++col;
	}
	else if(c==']')
	{
		fillToken(&tkn,c,row,col,"RSB");
		gotToken=1;
		++col;
	}
	else if(c=='}')
	{
		fillToken(&tkn,c,row,col,"RC");
		gotToken=1;
		++col;
	}
	else if(c=='+')
	{
		int d=fgetc(f1);
		if(d!='+')
		{
			fillToken(&tkn,c,row,col,"ARITHMETICOPERATOR");
			gotToken=1;
			++col;
			fseek(f1,-1,SEEK_CUR);
		}
		else
		{
			fillToken(&tkn,c,row,col,"UNARYOPERATOR");
			strcpy(tkn.lexeme,"++");
			gotToken=1;
			col+=2;
		}
	}
	else if(c=='-')
	{
		int d=fgetc(f1);
		if(d!='-')
		{
			fillToken(&tkn,c,row,col,"ARITHMETICOPERATOR");
			gotToken=1;
			++col;
			fseek(f1,-1,SEEK_CUR);
		}
		else
		{
			fillToken(&tkn,c,row,col,"UNARYOPERATOR");
			strcpy(tkn.lexeme,"--");
```

```
                gotToken=1;
                col+=2;
        }
}
else if(c=='=')
{
        int d=fgetc(f1);
        if(d!='=')
        {
                fillToken(&tkn,c,row,col,"ASSIGNMENTOPERATOR");
                gotToken=1;
                ++col;
                fseek(f1,-1,SEEK_CUR);
        }
        else
        {
                fillToken(&tkn,c,row,col,"RELATIONALOPERATOR");
                strcpy(tkn.lexeme,"==");
                gotToken=1;
                col+=2;
        }
}
else if(isdigit(c))
{
        tkn.row=row;
        tkn.col=col++;
        tkn.lexeme[0]=c;
        int k=1;
        while((c=fgetc(f1))!=EOF && isdigit(c))
        {
                tkn.lexeme[k++]=c;
                col++;
        }
        tkn.lexeme[k]='\0';
        strcpy(tkn.type,"NUMBER");
        gotToken=1;
        fseek(f1,-1,SEEK_CUR);
}
else if(c == '#')
{
        while((c = fgetc(f1)) != EOF && c != '\n');
        newLine();
}
else if(c=='\n')
{
        newLine();
        c = fgetc(f1);
        if(c == '#')
        {
                while((c = fgetc(f1)) != EOF && c != '\n');
                newLine();
        }
```

```
		else if(c != EOF)
		{
			fseek(f1, -1, SEEK_CUR);
		}
	}
	else if(isspace(c))
	{
		++col;
	}
	else if(isalpha(c)||c=='_')
	{
		tkn.row=row;
		tkn.col=col++;
		tkn.lexeme[0]=c;
		int k=1;
		while((c=fgetc(f1))!= EOF && isalnum(c))
		{
			tkn.lexeme[k++]=c;
			++col;
		}
		tkn.lexeme[k]='\0';
		if(isKeyword(tkn.lexeme))
		{
			strcpy(tkn.type,"KEYWORD");
		}
		else
		{
			strcpy(tkn.type,"IDENTIFIER");
		}
		gotToken=1;
		fseek(f1,-1,SEEK_CUR);
	}
	else if(c=='/')
	{
		int d=fgetc(f1);
		++col;//Do we check EOF here?
		if(d=='/')
		{
			while((c=fgetc(f1))!= EOF && c!='\n')
			{
				++col;
			}
			if(c=='\n')
			{
				newLine();
			}
		}
		else if(d=='*')
		{
			do
			{
				if(d=='\n')
```

```
            {
                newLine();
            }
            while((c==fgetc(f1))!= EOF && c!='*')
            {
                ++col;
                if(c=='\n')
                {
                    newLine();
                }
            }
            ++col;
        }while((d==fgetc(f1))!= EOF && d!='/' && (++col));

        ++col;
    }
    else
    {
        fillToken(&tkn,c,row,--col,"ARITHMETICOPERATOR");
        gotToken=1;
        fseek(f1,-1,SEEK_CUR);
    }
}
else if(c == '"')
{
    tkn.row = row;
    tkn.col = col;
    strcpy(tkn.type, "STRING LITERAL");
    int k = 1;
    tkn.lexeme[0] = '"';
    while((c = fgetc(f1)) != EOF && c != '"')
    {
        tkn.lexeme[k++] = c;
        ++col;
    }
    tkn.lexeme[k] = '"';
    gotToken = 1;
}
else if(c == '<' || c == '>' || c == '!')
{
    fillToken(&tkn, c, row, col, "RELATIONAL OPERATOR");
    ++col;
    int d = fgetc(f1);
    if(d == '=')
    {
        ++col;
        strcat(tkn.lexeme, "=");
    }
    else
    {
        if(c == '!')
        {
```

```
                        strcpy(tkn.type, "LOGICAL OPERATOR");
                    }
                    fseek(f1, -1, SEEK_CUR);
                }
                gotToken = 1;
            }
            else if(c == '&' || c == '|')
            {
                int d = fgetc(f1);
                if(c == d)
                {
                    tkn.lexeme[0] = tkn.lexeme[1] = c;
                    tkn.lexeme[2] = '\0';
                    tkn.row = row;
                    tkn.col = col;
                    ++col;
                    gotToken = 1;
                    strcpy(tkn.type, "LOGICAL OPERATOR");

                }
                else
                {
                    fseek(f1, -1, SEEK_CUR);
                }
                ++col;
            }
            else
            {
                ++col;
            }
    }
            return tkn;
    }



    int main()
    {
            FILE *f1=fopen("prog1InputFile.c","r");
            if(f1==NULL)
            {
                printf("Error! File cannot be opened!\n");
                return 0;
            }
            struct token tkn;
            while((tkn=getNextToken(f1)).row!=-1)
            {
                printf("<%s, %d, %d, %s>\n",tkn.lexeme,tkn.row,tkn.col,tkn.type);
            }
            fclose(f1);
    }
```



```
@lplab-Lenovo-Product: ~/Desktop/Sahil_180905048/lab3                                11:44 AM
student@lplab-Lenovo-Product:~/Desktop/Sahil_180905048/lab3$ gcc try.c -o try.out
student@lplab-Lenovo-Product:~/Desktop/Sahil_180905048/lab3$ ./try.out
<struct, 6, 1, IDENTIFIER>
<token, 6, 8, IDENTIFIER>
<{, 7, 1, LC>
<char, 8, 1, KEYWORD>
<lexeme, 8, 6, IDENTIFIER>
<64, 8, 13, NUMBER>
<;, 8, 16, SS>
<int, 9, 1, KEYWORD>
<row, 9, 5, IDENTIFIER>
<,, 9, 8, SS>
<col, 9, 9, IDENTIFIER>
<;, 9, 12, SS>
<char, 10, 1, KEYWORD>
<type, 10, 6, IDENTIFIER>
<20, 10, 11, NUMBER>
<;, 10, 14, SS>
<}, 11, 1, RC>
<;, 11, 2, SS>
<static, 12, 1, IDENTIFIER>
<int, 12, 8, KEYWORD>
<row, 12, 12, IDENTIFIER>
<=, 12, 15, ASSIGNMENTOPERATOR>
<1, 12, 16, NUMBER>
<,, 12, 17, SS>
<col, 12, 18, IDENTIFIER>
<=, 12, 21, ASSIGNMENTOPERATOR>
<1, 12, 22, NUMBER>
<;, 12, 23, SS>
<char, 13, 1, KEYWORD>
<buf, 13, 6, IDENTIFIER>
<2048, 13, 10, NUMBER>
<;, 13, 15, SS>
<const, 14, 1, KEYWORD>
<char, 14, 7, KEYWORD>
<specialsymbols, 14, 12, IDENTIFIER>
<=, 14, 28, ASSIGNMENTOPERATOR>
<{, 14, 29, LC>
<?, 14, 31, SS>
<,, 14, 33, SS>
<;, 14, 35, SS>
<,, 14, 37, SS>
<:, 14, 39, SS>
<,, 14, 41, SS>
<,, 14, 43, SS>
<}, 14, 45, RC>
<;, 14, 46, SS>
<const, 15, 1, KEYWORD>
<char, 15, 7, KEYWORD>
<*, 15, 12, ARITHMETIC OPERATOR>
```

//END///
/