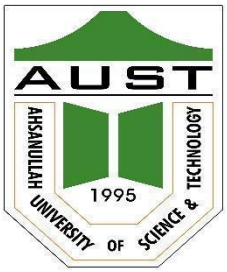

Ahsanullah University of Science & Technology

Department of Computer Science & Engineering

Course No : **CSE4130**
Course Title : **Formal Languages and Compilers Lab**
Assignment No : **02**

Date of Performance : **29-11-22**
Date of Submission : **13-12-22**

Submitted To : **Mr. Aminur Rahman & Mr. Al Hasib Mahamud**

Submitted By-
Group : **A_2**
Name : **Md. Nabil Rahman Khan**
Id : **190104044**
Section : **A**

Code:

```
/*Md Nabil Rahman Khan
190104044*/
#include <iostream>
#include <fstream>
#include <regex>
#include <string.h>
using namespace std;
string nilab="";
void filemaker2(string text1,string text2)
{
    nilab =nilab+" ["+text2+" "+text1+"] ";

}
string filemaker(string text,regex r,string text2)
{
    string result;
    regex_replace(back_inserter(result), text.begin(), text.end(),
            r, text2);
    return result;
}
bool isKeyword(string str)
                        //check if the given substring is a keyword
or not
{

    string keyword[]=
{"if","else","printf","while","for","do","break,","continue","int","dou
ble","float","return","char","case","long","short","typedef","switch
","unsigned","void","static","struct","sizeof","long","volatile","enu
m","const","bool","union","extern"};
    for(int k=0; k< (sizeof keyword / sizeof keyword[0]); k++)
    {

        if(keyword[k].compare(str)==0)return true;
    }
    return false;

}
bool isIdentifier(string str)
{
    regex e("([a-zA-Z_])[a-zA-Z0-9_]{0,31}");
    if (regex_match (str,e))
        return true;
    else return false;
}
bool isNum(string str)
{
    regex e("(\-)?(\\d+(?:\.\\d+)?)");
    if (regex_match (str,e))
        return true;
    else return false;
}
bool isOperand(string str)
                        //check if the given substring is a keyword
or not
{

    string Operand[]= {"+","-",
"*","/","++","--",">","<","=","<=",">=","!","!=","%","&&","|","||","==","
+=","/=","%=","&=","|=","^=",">>=","<<=",">=","<=","!="};
    for(int k=0; k< (sizeof Operand / sizeof Operand[0]); k++)
    {

        if(Operand[k].compare(str)==0)return true;
    }
    return false;
}
bool isSeparator(string str)
                        //check if the given substring is a keyword
or not
{

    string Separator[]= {",",";","\"","'"};
    for(int k=0; k< (sizeof Separator / sizeof Separator[0]); k++)
    {

        if(Separator[k].compare(str)==0)return true;
    }
    return false;

}
bool isPair(string str)
                        //check if the given substring is a keyword
or not
{

    string Pair[]= {"(",")","{","}","[","]"};
    for(int k=0; k< (sizeof Pair / sizeof Pair[0]); k++)
    {

        if(Pair[k].compare(str)==0)return true;
    }
    return false;

}
void regexprint(regex rgx,string text)
{
    string res="";
    for( sregex_iterator it(text.begin(), text.end(), rgx), it_end; it
!= it_end; ++it )
    {
        if(isKeyword((*it)[0].str()))filemaker2((*it)[0].str(),"Kw");
        else if(isIdentifier((*it)[0].str()))filemaker2((*it)[0].str(),"Id");
        else if(isNum((*it)[0].str()))filemaker2((*it)[0].str(),"Num");
        else
if(isOperand((*it)[0].str()))filemaker2((*it)[0].str(),"Op");
        else
if(isSeparator((*it)[0].str()))filemaker2((*it)[0].str(),"Sep");
        else if(isPair((*it)[0].str()))filemaker2((*it)[0].str(),"Par");
        else filemaker2((*it)[0].str(),"Unkown");

        //res=res +" " +(*it)[0].str();
    }
    //cout<< res;
}
int main()
{
    fstream file;
    string  text,temp;
    file.open("190104044_input.txt", ios::in | ios::app);
    if (!file.is_open())cout << "No FIle ! Error";
    else
    {
        while (!file.eof())
        {
            getline(file, temp);
            text = text  + temp + "\n";
        }
        regex r0("printf(.*)\\((.*)\\);");
        text = filemaker(text, r0,"printf ( \" \" );");
        cout<<text;

        string operator1[]= { "\\;", "\\+", "\\-","\\'",
"\\*","\\/","\\%","\\,","\\=","\\>","\\<","\\&&","\\|","\\!","\\(","\\)","\\&" };
        for(int k=0; k< (sizeof operator1 / sizeof operator1[0]);
```

```
k++)
	{
	    string l=" "+operator1[k].substr(1,operator1[k].size())+" ";
	    regex r1(operator1[k]);
	    text = filemaker(text, r1,l);
	}
	string operator2[] = {"(\\+)( )*(\\+)","(\\=)( )*(\\=)","(\\+)(
)*(\\=)","(\\-)( )*(\\=)","(\\*)( )*(\\=)","(\\/)( )*(\\=)","(\\%)(
)*(\\=)","(\\&)( )*(\\=)","(\\|)( )*(\\=)","(\\^)( )*(\\=)","(\\>)( )*(\\>)(
)*(\\=)","(\\<)( )*(\\<)( )*(\\=)","(\\>)( )*(\\=)","(\\<)( )*(\\=)","(\\!)(
)*(\\=)"};
	string f[]=
{"++","==","+=","-=","*=","/=","%=","&=","|=","^=",">>=","<<=",">=
","<=","!="};
	for(int k=0; k< (sizeof operator2 / sizeof operator2[0]); k++)
	{
	    regex r1(operator2[k]);
	    text = filemaker(text, r1,f[k]);
	}
	//DIVING INTO WORDS

	regex r2("(\\S+)");
	regexprint(r2,text);
	cout<<nilab;


	file.close();
	file.open("OUTPUTIAB2.txt", ios::out | ios::trunc);
	file << nilab;
	file.close();

   }
}
```

Input.txt

```
char c; int x1, x_2; float y1, y2; *x1=5; x_2=
10; y1=2.5+x1*45; y2=100.o5-x_2/3;
if(y1<=y2) c='y'; else c='n';for( int i=0;i<30;i++
) { printf("Hello") ;}
```

Output.txt

```
 [Kw char]  [Id c]  [Sep ;]  [Kw int]  [Id x1]  [Sep ,]  [Id x_2]  [Sep
;]  [Kw float]  [Id y1]  [Sep ,]  [Id y2]  [Sep ;]  [Op *]  [Id x1]  [Op
=]  [Num 5]  [Sep ;]  [Id x_2]  [Op =]  [Num 10]  [Sep ;]  [Id y1]
[Op =]  [Num 2.5]  [Op +]  [Id x1]  [Op *]  [Num 45]  [Sep ;]  [Id
y2]  [Op =]  [Unkown 100.o5]  [Op -]  [Id x_2]  [Op /]  [Num 3]
[Sep ;]  [Kw if]  [Par (]  [Id y1]  [Op <=]  [Id y2]  [Par )]  [Id c]
[Op =]  [Sep ']  [Id y]  [Sep ']  [Sep ;]  [Kw else]  [Id c]  [Op =]
[Sep ']  [Id n]  [Sep ']  [Sep ;]  [Kw for]  [Par (]  [Kw int]  [Id i]
[Op =]  [Num 0]  [Sep ;]  [Id i]  [Op <]  [Num 30]  [Sep ;]  [Id i]
[Op ++]  [Par )]  [Par {]  [Kw printf]  [Par (]  [Sep "]  [Sep "]  [Par
)]  [Sep ;]  [Par }]
```

Question:

Suppose, we have a C source program scanned and filtered as it was done in Session 1. We now take that modified file as input, and separate the lexemes first. We further recognize and mark the lexemes as different types of tokens like keywords, identifiers, operators, separators, parenthesis , numbers, etc.