VATSAL SHAH
TE4-59

EXPERIMENT 2

CODE:

```
%{
int ch=0, bl=0, ln=0,
wr=0 %}

%%
[/n]{ln++;wr++}
[\t]{bl++;wr++}
[“”]{bl++;wr++}
[^\n\t]{ch++;}
%%
Int main()
{
FILE *fp;
Char file[10];
Printf(“Enter file name”);
Scanf(“%s”,file);
Yyin=fp;
Yylex();
Printf(“character=%d \n Blank=%d \n Lines=%d \n Words=%d”, ch, bl, ln, wr);
Return 0; }
```

OUTPUT:

```
$lex prog.l
$cc lex.yy.c –ll
$a.out
Enter File Name:
sample Character=16
Blank=2
Lines= 2
Words = 3
```

VATSAL SHAH
TE4-59

EXPERIMENT 3A

CODE:

```
 %{
#include <stdio.h>;
%}
%%

if|else|while|int|switch|for|char

          {printf("keyword");}
[a-z]([a- z]|[0-9])*
          {printf("identifier");}
[0-9]*
          {printf("number");}
.*
          {printf("invalid");}
%%

main()
{
          yylex( );
          return 0;
}
          int yywrap()
{
}
```

OUTPUT:

```
A
Identifier
0
Number
if
keyword
```

VATSAL SHAH

TE4-59

EXPERIMENT 3B

CODE:

```
spe=["#","<",">",";","@","_",","]
oper=['+','-','*','/','%','=','!',"++"]
key=["int","float","char","double","bool","void","extern","unsigned","goto","static","class","
struct","for",
"if","else","return","register","long","while","do","printf","scanf"]
predirect=["include","define"]
header=["stdio.h","conio.h","malloc.h","process.h","string.h","ctype.h"]
bracket=["(",")","[","]","{","}"]
quote=['"']
m=input("Enter file name:")
f=open(m,"r")
m_lines=0
for line in f:
    words = line.split()
    m_lines += 1
    print("\nLINE",m_lines)
    token=0
    for i in words:
        if(i in spe):
            print(i,"is special character")
            token =token+ 1
        elif(i in oper):
            print(i,"is Operator")
            token =token+ 1
        elif(i in quote):
            print(i,"is Quote")
            token =token+ 1
        elif(i in key):
            print(i,"is keyword")
            token =token+ 1
        elif(i in predirect):
            print(i,"is Pre-Processor")
            token =token+ 1
        elif(i in header):
            print(i,"is Header")
            token =token+ 1
        elif(i in bracket):
            print(i,"is Bracket")
            token =token+ 1
        elif((i>='a' and i<= 'z') or (i>='A' and i<='Z')):
            print(i, "is an identifier")
            token =token+ 1
        elif(i>='0' and i<= '9'):
            print(i, "is number")
            token =token+ 1
    print("Number of tokens are",token)
```

INPUT FILE: abc_vatsal.txt

```
# include < stdio.h >
# include < conio.h >
void main ( )
{
int a = 15 ;
float b=25.5;
a ++ ;
printf ( ' Value of a is ' , a ) ;
}
```

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp3b.py =================
Enter file name:abc_vatsal.txt

LINE 1
# is special character
include is Pre-Processor
< is special character
stdio.h is Header
> is special character
Number of tokens are 5

LINE 2
# is special character
include is Pre-Processor
< is special character
conio.h is Header
> is special character
Number of tokens are 5

LINE 3
void is keyword
main is an identifier
( is Bracket
) is Bracket
Number of tokens are 4

LINE 4
{ is Bracket
```

VATSAL SHAH

TE4-59

```
Number of tokens are 1

LINE 5
int is keyword
a is an identifier
= is Operator
15 is number
; is special character
Number of tokens are 5

LINE 6
float is keyword
b=25.5; is an identifier
Number of tokens are 2

LINE 7
a is an identifier
++ is Operator
; is special character
Number of tokens are 3

LINE 8
printf is keyword
( is Bracket
' is Quote
Value is an identifier
of is an identifier
a is an identifier
is is an identifier
' is Quote
, is special character
a is an identifier
) is Bracket
; is special character
Number of tokens are 12

LINE 9
} is Bracket
Number of tokens are 1
>>>
```

VATSAL SHAH

TE4-59

EXPERIMENT 4

CODE:

```
import re
class MOT:
    def __init__(self,mnemonic,binaryop,insLength,insFormat):
        self.mnemonic = mnemonic
        self.binaryop = binaryop
        self.insLength = insLength
        self.insFormat = insFormat
Mlist = []
Mlist1 = []
Mlist.append(MOT('L','58','10','001'))
Mlist.append(MOT('A','5A','10','001'))
Mlist.append(MOT('ST','50','10','001'))
class POT:
    def __init__(self,psop,address):
        self.psop = psop
        self.address = address

Plist = []
Plist1 = []
Plist.append(POT('START','P1START'))
Plist.append(POT('USING','P1USING'))
Plist.append(POT('DC','P1DC'))
Plist.append(POT('DS','P1DS'))
Plist.append(POT('END','P1END'))

class ST:
    def __init__(self,symbol,value,length,relocation):
        self.symbol = symbol
        self.value = value
        self.length = length
        self.relocation = relocation
STlist=[]
def remove_values_from_list(the_list, val):
    return [value for value in the_list if value != val]

f = open("input_vatsal.txt", "rt")
addr = 0
for line in f:
    s = re.split(" |\t|\n",line)
    s=remove_values_from_list(s,"")
    print(s)
    if(len(s) == 2):
        operands = s[1].split(',')
        if(s[0]!='USING'):
            addr+=4
        for item in Plist:
            if(item.psop == s[0]):
                Plist1.append(POT(s[0],item.address))
        for item in Mlist:
            if(item.mnemonic == s[0]):
                Mlist1.append(MOT(s[0],item.binaryop,item.insLength,item.insFormat))
    else:
        if(len(s)==3):
            if(s[1]=='START'):
                STlist.append(ST(s[0],str(hex(addr)),0,'R'))
            else:
                STlist.append(ST(s[0],str(hex(addr)),4,'R'))
            if(s[1]!='START'):
                addr+=4
            for i in range(len(s)):
                for item in Plist:
                    if(item.psop == s[i]):
                        Plist1.append(POT(s[i],item.address))
                for item in Mlist:
                    if(item.mnemonic == s[i]):
                        Mlist1.append(MOT(s[i],item.binaryop,item.insLength,item.insFormat))

print("SYMBOL TABLE")
for item in STlist:
    print(item.symbol+"\t\t"+item.value+"\t\t"+str(item.length)+"\t\t"+item.relocation+"\t\t")

print("PSEUDO OPERATION TABLE")
for item in Plist1:
    print(item.psop+"\t\t"+item.address)

print("MACHINE OPERATION TABLE")
for item in Mlist1:
    print(item.mnemonic+"\t\t"+item.binaryop+"\t\t"+item.insLength+"\t\t"+item.insFormat)
```

INPUT FILE: input_vatsal.txt

```
JOHN START 0
USING *,15
L 1,FIVE
A 1,FOUR
ST 1,TEMP
FOUR Dc F'4'
FIVE DC F'5'
TEMP DS 1F
END
```

VATSAL SHAH

TE4-59

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp4.py ==================
['JOHN', 'START', '0']
['USING', '*,15']
['L', '1,FIVE']
['A', '1,FOUR']
['ST', '1,TEMP']
['FOUR', 'Dc', "F'4'"]
['FIVE', 'DC', "F'5'"]
['TEMP', 'DS', '1F']
['END']
SYMBOL TABLE
JOHN        0x0         0           R
FOUR        0xc         4           R
FIVE        0x10        4           R
TEMP        0x14        4           R
PSEUDO OPERATION TABLE
START               P1START
USING               P1USING
DC          P1DC
DS          P1DS
MACHINE OPERATION TABLE
L           58          10          001
A           5A          10          001
ST          50          10          001
>>>
```

VATSAL SHAH
TE4-59

EXPERIMENT 5

CODE:

```
class MDT():
    def __init__(self,index,card):
        self.index = index
        self.card = card
    def __repr__(self):
        return ""+str(self.index)+"\t"+self.card

class MNT():
    def __init__(self,index,card,mdtindex):
        self.index = index
        self.card = card
        self.mdtindex = mdtindex
    def __repr__(self):
        return ""+str(self.index)+"\t"+self.card+"\t"+str(self.mdtindex)

class ALA():
    def __init__(self,index_marker,args):
        self.index_marker = index_marker
        self.args = args
    def __repr__(self):
        return ""+str(self.index_marker)+"\t\t"+self.args

def remove_values_from_list(the_list, val):
    return [value for value in the_list if value != val]
import re
indexmnt = 0
MNT_list = []
ALA_list = []
indexala = 0
MDT_list = []
done = False
line_list = []

if __name__ == '__main__':
    f = open("input_5_vatsal.txt", "rt")
    addr = 0
    index = 0
    for line in f:
        s = re.split(" |\t|\n",line)
        s=remove_values_from_list(s,"")
        line_list.append(s)
    args_list = []
    for i,line in enumerate(line_list):
        wordString = ""
        for word in line:
            if("&" in word):
                if("," not in word and (word not in args_list)):
                    ALA_list.append(ALA(indexala,word))
                    indexala+=1
                    args_list.append(word)
            wordString+=word+" "
            if(word == 'PROG'):
                done = True
                break
        if done == True:
            break
        if "MACRO" not in wordString:
            MDT_list.append(MDT(index,wordString))
            index = index+1
        if "MACRO" in line_list[i-1]:
            MNT_list.append(MNT(indexmnt,line_list[i][0],index-1))
            indexmnt+=1
    print("MDT Table")
    print("index\tcard")
    print(*MDT_list,sep="\n")
    print()
    print("MNT Table")
    print("index\tcard\tmdtindx")
    print(*MNT_list,sep ="\n")
    print()
    print("ALA Table")
    print("index_marker\tArguments")
    print(*ALA_list,sep ="\n")
```

INPUT FILE: input_5_vatsal.txt

```
MACRO
XYZ &a
ST 1,&a
MEND
MACRO
MIT &z
MACRO
&z &w
AR 4,&w
XYZ ALL
MEND
ST &w,ALL
```

```
MEND
PROG START
USING *,15
MIT HELLO
ST 2,3
HELLO YALE
YALE EQU 5
ALL DC f'3'
END
```

OUTPUT:

```
================== RESTART: C:/Vatsal/Intel/Desktop/exp5.py ==================
MDT Table
index   card
0       XYZ &a
1       ST 1,&a
2       MEND
3       MIT &z
4       &z &w
5       AR 4,&w
6       XYZ ALL
7       MEND
8       ST &w,ALL
9       MEND

MNT Table
index   card    mdtindx
0       XYZ     0
1       MIT     3
2       &z      4

ALA Table
index_marker  Arguments
0             &a
1             &z
2             &w
```

VATSAL SHAH
TE4-59

EXPERIMENT 6

CODE:

```
count=0
temp = ['X','Y', 'Z']
n=int(input("Enter number of productions to be entered:"))
print (n)
while(count < n):
    left=input("Enter left hand side of production" + str(count)+ " : ")
    right=input("Enter right hand side of production" + left+ " :" )
    if (right[0] == left):
        print("Left Recursion Present")
        str1 = right.split('|')
        alpha = str1[0]
        beta = str1[1]
        print(beta + temp[count])
        print(alpha[1:] + temp[count] + '|9')
    else:
        print("Left Recursion Not Present")
    count=count+1
```

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp6.py ==================
Enter number of productions to be entered:3
3
Enter left hand side of production0 : E
Enter right hand side of productionE :E+T|T
Left Recursion Present
TX
+TX|9
Enter left hand side of production1 : T
Enter right hand side of productionT :T*F|F
Left Recursion Present
FY
*FY|9
Enter left hand side of production2 : F
Enter right hand side of productionF :(E)|i
Left Recursion Not Present
>>>
```

VATSAL SHAH

TE4-59

EXPERIMENT 7

CODE:

```
NT = []
T = []
temp = []
P = {}
first1 = {}
t1 = []

n_NT = int(input("Enter number of Non-terminals : "))
n_T = int(input("Enter number of Terminals : "))

print("Enter List of Non-terminals")
for i in range(0, n_NT):
    item = input()
    NT.append(item)

print("Enter List of Terminals")
print("Enter 9 for epsilon")
for i in range(0, n_T):
    item = input()
    T.append(item)

print("Enter Production")
for i in range(0,n_NT):
    print("Enter production for" + NT[i])
    ele = input()
    P[NT[i]]=ele


print(NT)
print(T)
print(P)
n_P = len(P)


for i in range(n_NT):
    nonter=NT[i]
    pro = (P.get(nonter))

    if '|' in pro:
        str1 = pro.split('|')
        l = len(str1)
        for i in range(l):
            t = str1[i]
            if(t[0] in T):
                temp.append(t[0])
            elif(t[0] in NT):
                print("NT")
                s = t[0]
                s1 = first1.get(s)
                temp.append(s1)
            else:
                print("Not")

    else:
        if(pro[0] in T):
            temp.append(pro[0])
        elif(pro[0] in NT):
            print("NT")
            sNT = pro[0]
            sNT1 = first1.get(sNT)
            temp.append(sNT1)
        else:
            print("Not")
    first1[nonter] = temp
    temp =[]


for i in range(len(first1)):
    check = first1.get(NT[i])
    if (None in check):
        t1.append(NT[i])


for k in range(len(t1)):
    p1 = P.get(t1[k])
    check = p1[0]
    if(check in t1):
        pro1 = P.get(check)
        pro1_ch = pro1[0]
        value = first1.get(pro1_ch)
        first1[t1[k]] = value
    else:
        value = first1.get(check)
        first1[t1[k]]=value


print("FIRST")
print(first1)
```

VATSAL SHAH

TE4-59

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp7.py ==================
Enter number of Non-terminals : 5
Enter number of Terminals : 6
Enter List of Non-terminals
S
X
T
Y
F
Enter List of Terminals
Enter 9 for epsilon
+
*
(
)
i
9
Enter Production
Enter production forS
TX
Enter production forX
+TX|9
Enter production forT
FY
Enter production forY
*FY|9
Enter production forF
(S)|i
['S', 'X', 'T', 'Y', 'F']
['+', '*', '(', ')', 'i', '9']
{'S': 'TX', 'X': '+TX|9', 'T': 'FY', 'Y': '*FY|9', 'F': '(S)|i'}
NT
NT
FIRST
{'S': ['(', 'i'], 'X': ['+', '9'], 'T': ['(', 'i'], 'Y': ['*', '9'], 'F': ['(', 'i']}
>>>
```

VATSAL SHAH

TE4-59

EXPERIMENT 8

CODE:

```
NT = []
T = []
temp = []
temp_follow = []
P = {}
first1 = {}
follow = {}
t1 = []
start='S'

n_NT = int(input("Enter number of Non-terminals : "))
n_T = int(input("Enter number of Terminals : "))


print("Enter List of Non-terminals")
for i in range(0, n_NT):
    item = input()
    NT.append(item)
print("Enter List of Terminals")
print("Enter 9 for epsilon")
for i in range(0, n_T):
    item = input()
    T.append(item)


print("Enter Production")
for i in range(0,n_NT):
    print("Enter production for" + NT[i])
    ele = input()
    P[NT[i]]=ele
print(NT)
print(T)
print(P)
n_P = len(P)


for i in range(n_NT):
    nonter=NT[i]
    pro = (P.get(nonter))
    if '|' in pro:
        str1 = pro.split('|')
        l = len(str1)
        for i in range(l):
            t = str1[i]
            if(t[0] in T):
                temp.append(t[0])
            elif(t[0] in NT):
                print("NT")
                s = t[0]
                s1 = first1.get(s)
                temp.append(s1)
            else:
                print("Not")
    else:
        if(pro[0] in T):
            temp.append(pro[0])
        elif(pro[0] in NT):
            print("NT")
            sNT = pro[0]
            sNT1 = first1.get(sNT)
            temp.append(sNT1)
        else:
            print("Not")

    first1[nonter] = temp
    temp =[]

for i in range(len(first1)):
    check = first1.get(NT[i])
    if (None in check):
        t1.append(NT[i])


for k in range(len(t1)):
    p1 = P.get(t1[k])
    check = p1[0]
    if(check in t1):
        pro1 = P.get(check)
        pro1_ch = pro1[0]
        value = first1.get(pro1_ch)
        first1[t1[k]] = value
    else:
        value = first1.get(check)
        first1[t1[k]]=value

print(first1)
```

```
for i in range(n_NT):
    s = NT[i]
    print(s)
    if(s == start):
        temp_follow.append('$')
        follow[s] = temp_follow
    for j in range(n_NT):
        pro = (P.get(NT[j]))
        if(s in pro):
            pos = pro.find(s)
            next1 = pos+1
            if(next1 == (len(pro))):
                print("its last")
            else:
                next_ch = pro[next1]
            if(next_ch in T):
                temp_follow.append(next_ch)
                follow[s] = temp_follow
            elif(next_ch in NT):
                 check = first1.get(next_ch)
                 if('9' in check):
                    check.remove('9')
                    add_lhs = follow[NT[j]]
                    check.extend(add_lhs)
                    follow[s] = check
            else:
                add_lhs = follow.get(NT[j])
                follow[s] = add_lhs
    next_ch=""

print(follow)
```

OUTPUT:

```
================== RESTART: C:/Vatsal/Intel/Desktop/exp8.py ==================
Enter number of Non-terminals : 5
Enter number of Terminals : 6
Enter List of Non-terminals
S
X
T
Y
F
Enter List of Terminals
Enter 9 for epsilon
+
```

VATSAL SHAH

TE4-59

```
*
(
)
i
9
Enter Production
Enter production forS
TX
Enter production forX
+TX|9
Enter production forT
FY
Enter production forY
*FY|9
Enter production forF
(S)|i
['S', 'X', 'T', 'Y', 'F']
['+', '*', '(', ')', 'i', '9']
{'S': 'TX', 'X': '+TX|9', 'T': 'FY', 'Y': '*FY|9', 'F': '(S)|i'}
NT
NT
{'S': ['(', 'i'], 'X': ['+', '9'], 'T': ['(', 'i'], 'Y': ['*', '9'], 'F': ['(', 'i']}
S
X
its last
T
Y
its last
F
{'S': ['$', ')'], 'X': ['$', ')'], 'T': ['+', '$', ')'], 'Y': ['+', '$', ')'], 'F': ['*', '+', '$', ')']}
>>>
```

VATSAL SHAH
TE4-59

EXPERIMENT 9A

CODE:

```
ans = 'y'
line = []
temp = []
def isNumber(s) :
    for i in range(len(s)) :
        if s[i].isdigit() != True :
            return False
    return True


def isFloat(s):
    try :
        float(s)
        return True
    except :
        return False


while(ans == 'y'):
    t1 = input("Enter the instruction")
    line.append(t1)
    ans = input("Do you want to continue")
print(line)
l = len(line)


for i in range(l):
    if '=' in line[i]:
        str1 = line[i].split('=')
    if isNumber(str1[1]) or isFloat(str1[1]):
        no = str1[1]
        for k in range(l):
            if '=' in line[k]:
                s1 = line[k].split('=')
                if str1[0] in s1[1]:
                    subs = [str1[0], no]
                    line[k]=line[k].replace(subs[0], subs[1])
                    temp.append(line[i])


for j in range(len(temp)):
    t1 = temp[j]
    line.remove(t1)
```

```
print(line)
```

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp9.py =================
Enter the instructionp=3.14
Do you want to continuey
Enter the instructiona=p*r*r
Do you want to continuen
['p=3.14', 'a=p*r*r']
['a=3.14*r*r']
>>>
```

VATSAL SHAH
TE4-59

EXPERIMENT 9B

CODE:

```
program = ['a=b*c', 'x=a', 'd=b*c+15']
op = ['+', '*', '-', '/']
final = []
flag = 'False'
for i in range(len(program)):
    temp = program[i]
    if('=' in temp):
        str1 = temp.split('=')
        check = str1[1]
        res = any(ele in check for ele in op)
        flag = str(res)
        if(flag == 'True'):
            final.append(temp)


print("Program before Dead Code Elimination")
print(program)
print("Program After Dead Code Eminination")
print(final)
```

OUTPUT:

```
================= RESTART: C:/Vatsal/Intel/Desktop/exp10.py =================
Program before Dead Code Elimination
['a=b*c', 'x=a', 'd=b*c+15']
Program After Dead Code Eminination
['a=b*c', 'd=b*c+15']
>>>
```