G.Vikas

411924

Compiler Design
Lab-Record

G.vikas

411924

Code:

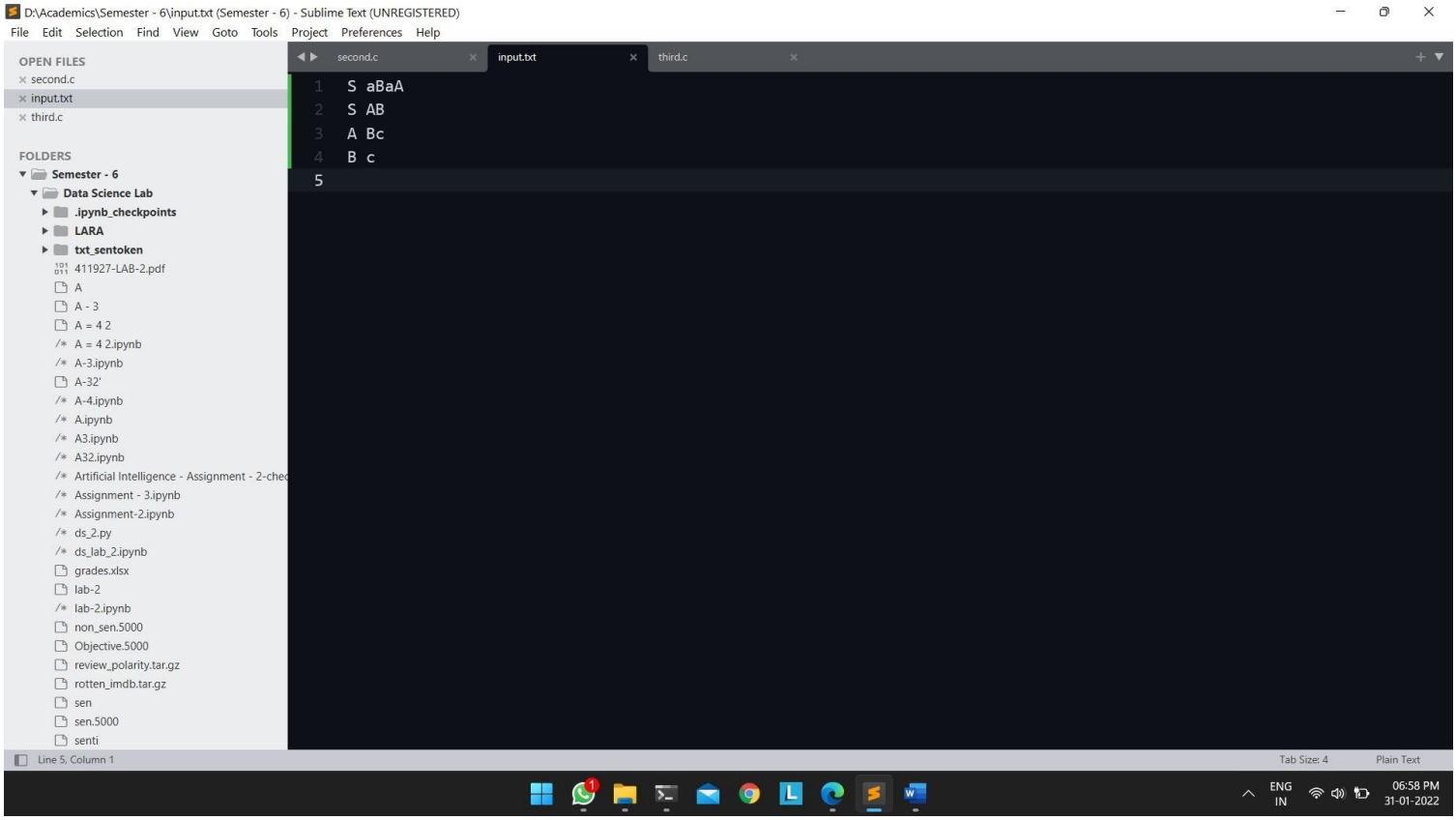
```
#include<stdio.h>
#include<string.h> #include<conio.h>
int i,j,k,l,m,n=0,o,p,nv,z=0,t,x=0; char
str[10],temp[20],temp2[20],temp3[20];

struct prod
{
 char lhs[10],rhs[10][10];
int n;
}pro[10];
void findter()
{
 for(k=0;k<n;k++)
 {
 if(temp[i]==pro[k].lhs[0])
 {
 for(t=0;t<pro[k].n;t++)
 {
 for(l=0;l<20;l++)
temp2[l]='\0';
for(l=i+1;l<strlen(temp);l++)
temp2[l-i-1]=temp[l];
for(l=i;l<20;l++) temp[l]='\0';
 for(l=0;l<strlen(pro[k].rhs[t]);l++)

temp[i+l]=pro[k].rhs[t][l]; strcat(temp,temp2);
 if(str[i]==temp[i]) return;
```

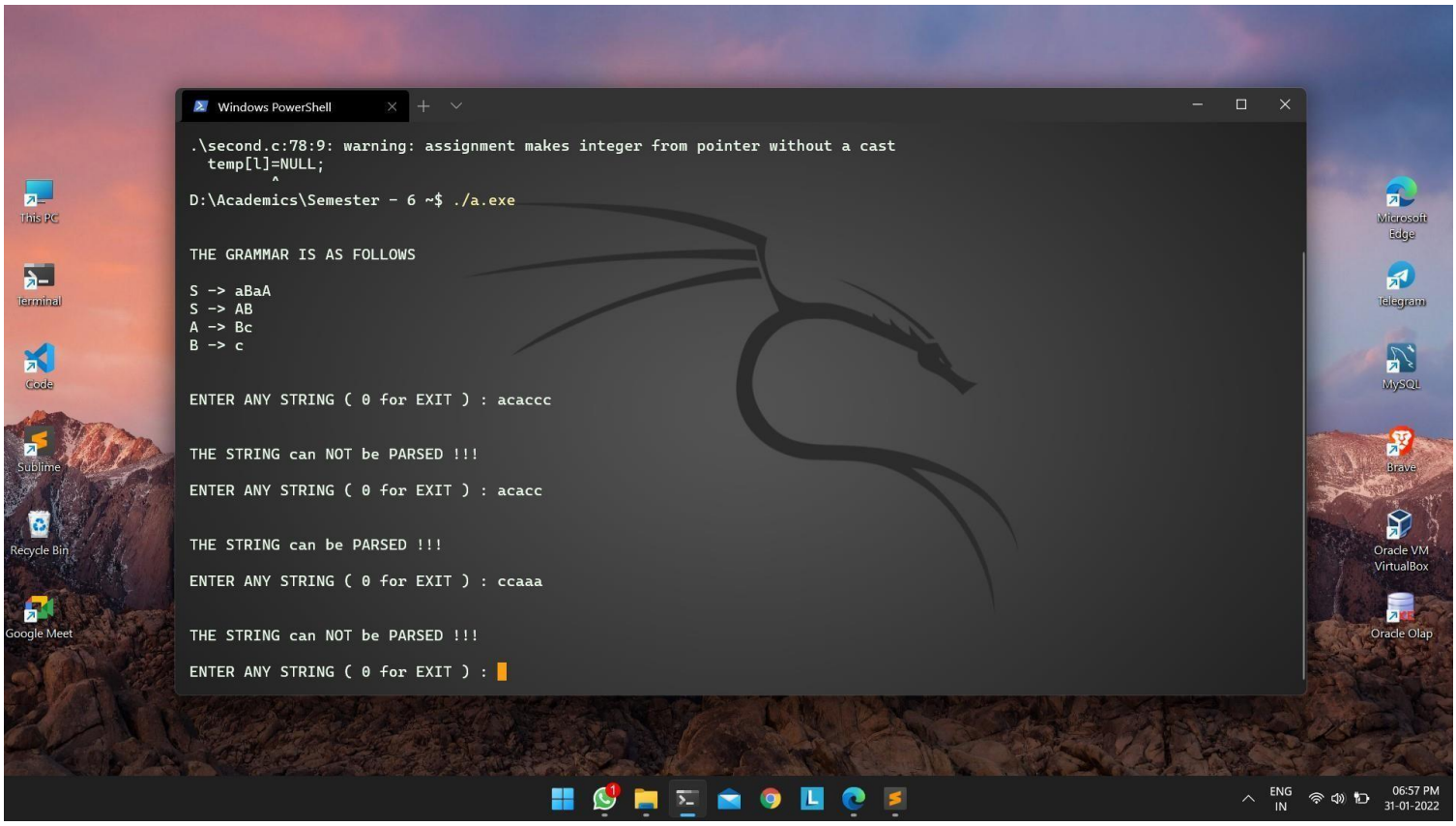
```
else if(str[i]!=temp[i] && temp[i]>=65 && temp[i]<=90) break;
}
break;
```

```
 }
 }
 if(temp[i]>=65 && temp[i]<=90)
 findter();
}

void main()
{
 FILE *f;
for(i=0;i<10;i++)
pro[i].n=0;
f=fopen("input.txt","r");
while(!feof(f))
 {
 fscanf(f,"%s",pro[n].lhs);
if(n>0)
 {
 if( strcmp(pro[n].lhs,pro[n-1].lhs) == 0 )
 {
 pro[n].lhs[0]='\0';
 fscanf(f,"%s",pro[n-1].rhs[pro[n-1].n]);
 pro[n-1].n++;
 continue;
 }
 }
 fscanf(f,"%s",pro[n].rhs[pro[n].n]);
 pro[n].n++;
 n++;
 } n--;
 printf("\n\nTHE GRAMMAR IS AS FOLLOWS\n\n");
 for(i=0;i<n;i++)
 for(j=0;j<pro[i].n;j++)
```

```
 printf("%s -> %s\n",pro[i].lhs,pro[i].rhs[j]);
 while(1)
 {
 for(l=0;l<10;l++)
str[0]=NULL;
 printf("\n\nENTER ANY STRING ( 0 for EXIT ) : ");
scanf("%s",str); if(str[0]=='0') exit(1);
for(j=0;j<pro[0].n;j++)
 {
 for(l=0;l<20;l++)
temp[l]=NULL;
strcpy(temp,pro[0].rhs[j]); m=0;
for(i=0;i<strlen(str);i++)
 {
 if(str[i]==temp[i])
 m++;
 else if(str[i]!=temp[i] && temp[i]>=65 && temp[i]<=90)
 {
 findter();
if(str[i]==temp[i])
m++;
 }
 else if( str[i]!=temp[i] && (temp[i]<65 || temp[i]>90) )
break;
 }
 if(m==strlen(str) && strlen(str)==strlen(temp))
 {
 printf("\n\nTHE STRING can be PARSED !!!");
 break;
 }
 }
```

```
if(j==pro[0].n) printf("\n\nTHE STRING can NOT
be PARSED !!!");
}
getch();
}
```



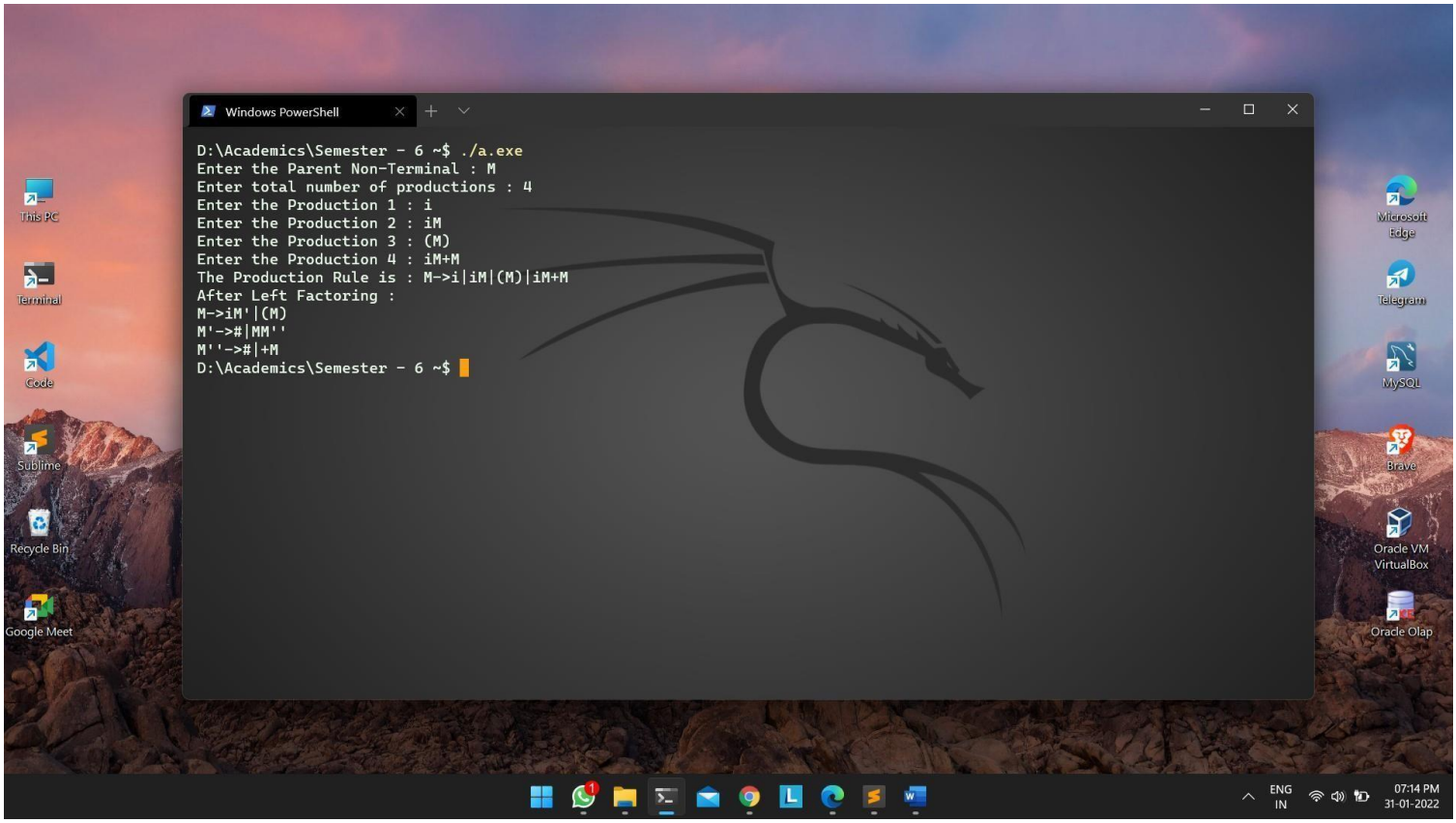


Code:

```
#include <iostream>
#include <string> using
namespace std; int
main()
{ int n,j,l,i,m; int
len[10] = {}; string
a, b1, b2, flag; char
c;
 cout << "Enter the Parent Non-Terminal : ";
cin >> c;
 a.push_back(c); b1
+= a + "\'->";  b2
+= a + "\'\'->";; a
+= "->";
 cout << "Enter total number of productions : ";
cin >> n;
```

```
 for (i = 0; i < n; i++)
 {
 cout << "Enter the Production " << i + 1 << " : ";
cin >> flag; len[i] = flag.size(); a += flag;
 if (i != n - 1)
 {
 a += "|";
}
}
 cout << "The Production Rule is : " << a << endl;
char x = a[3];
 for (i = 0, m = 3; i < n; i++)
 {
 if (x != a[m])
 {
 while (a[m++] != '|');
 }
 else
 {
 if (a[m + 1] != '|')
 {
 b1 += "|" + a.substr(m + 1, len[i] - 1);
a.erase(m - 1, len[i] + 1);
 }
 else
 {
 b1 += "#";
 a.insert(m + 1, 1, a[0]);
 a.insert(m + 2, 1, '\'');
 m += 4;
 }
 }
```

```
 }
 char y = b1[6];
 for (i = 0, m = 6; i < n - 1; i++)
 {
 if (y == b1[m])
 {
 if (b1[m + 1] != '|')
 {
 flag.clear();
 for (int s = m + 1; s < b1.length(); s++)
 {
 flag.push_back(b1[s]);
}
 b2 += "|" + flag;
 b1.erase(m - 1, flag.length() + 2);
 }
 else
 {
 b1.insert(m + 1, 1, b1[0]);
b1.insert(m + 2, 2, '\''); b2
+= "#";
 m += 5;
 }
 }
 }
 b2.erase(b2.size() - 1);
 cout << "After Left Factoring : " << endl;
cout << a << endl; cout << b1 << endl;
cout << b2 << endl;
 return 0;
}
```



Code:

```
#include<stdio.h>
#include<string.h> #define SIZE
10
int main ()
{
 char non_terminal;
 char beta,alpha;
 int num,i;
 char production[10][SIZE];
 int index=3;
 printf("Enter Number of Production : ");   scanf("%d",&num);
 printf("Enter the grammar as E->E-A :\n");   for(i=0;i<num;i++){
 scanf("%s",production[i]);
 }
```

```
 for(i=0;i<num;i++){
 printf("\nGRAMMAR : : : %s",production[i]);
non_terminal=production[i][0];
if(non_terminal==production[i][index]) {
alpha=production[i][index+1];   printf(" is left
recursive.\n");   while(production[i][index]!=0 &&
production[i][index]!='|')   index++;
if(production[i][index]!=0) {
beta=production[i][index+1];
 printf("Grammar without left recursion:\n");   printf("%c-
>%c%c\'",non_terminal,beta,non_terminal);   printf("\n%c\'-
>%c%c\'|E\n",non_terminal,alpha,non_terminal);
 }
 else
 printf(" can't be reduced\n");
 }
 else
 printf(" is not left recursive.\n");   index=3;
 }   }
```

D:\Academics\Semester - 6\fourth.c (Semester - 6) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

second.c | input.txt | third.cpp | fourth.c

```
beta=production[i][index+1];
```

Windows PowerShell

```
Enter the grammar as E->E-A :
E->EA|A
A->At|a
D:\Academics\Semester - 6 ~$ ./a.exe
Enter Number of Production : 4
Enter the grammar as E->E-A :
E->EA|A
A->At|a
D:\Academics\Semester - 6 ~$ ./a.exe
Enter Number of Production : 4
Enter the grammar as E->E-A :
E->EA|A
A->AT|a
T->a
E->i

GRAMMAR : : : E->EA|A is left recursive.
Grammar without left recursion:
E->AE'
E'->AE'|E

GRAMMAR : : : A->AT|a is left recursive.
Grammar without left recursion:
A->aA'
A'->TA'|E

GRAMMAR : : : T->a is not left recursive.

GRAMMAR : : : E->i is not left recursive.
D:\Academics\Semester - 6 ~$
```

Line 30, Column 4 | Spaces: 4 | C++

ENG IN 07:25 PM 31-01-2022

Windows PowerShell

```
D:\Academics\Semester - 6 ~$ ./a.exe
Enter Number of Production : 4
Enter the grammar as E->E-A :
E->E+T|T
S->S0S1S|01
D:\Academics\Semester - 6 ~$ ./a.exe
Enter Number of Production : 2
Enter the grammar as E->E-A :
E->E+T|T
S->S0S1S|01

GRAMMAR : : : E->E+T|T is left recursive.
Grammar without left recursion:
E->TE'
E'->+E'|E

GRAMMAR : : : S->S0S1S|01 is left recursive.
Grammar without left recursion:
S->0S'
S'->0S'|E
D:\Academics\Semester - 6 ~$
```

ENG IN 07:28 PM 31-01-2022

Code:

```
%{
int n = 0 ;
%}

%%

"while"|"if"|"else" {n++;printf("\t keywords : %s", yytext);}

"int"|"float" {n++;printf("\t keywords : %s", yytext);}

[a-zA-Z_][a-zA-Z0-9_]* {n++;printf("\t identifier : %s", yytext);}

"<="|"=="|"="|"++"|"-"|"*"|"+" {n++;printf("\t operator : %s", yytext);}

[(){}|, ;] {n++;printf("\t separator : %s", yytext);}

[0-9]*"."[0-9]+ {n++;printf("\t float : %s", yytext);}

[0-9]+ {n++;printf("\t integer : %s", yytext);}

. ;

%%
int main()
{

  yylex();

  printf("\n total no. of token = %d\n", n);
}
```

int yywrap(){return(1);}

Kali-Linux-2020.2a-vbox-amd64 [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

kali@kali: ~/Desktop/La... 09:13 AM 84%

Click to start dragging "kali@kali: ~/Desktop/Language-Processors-Lab/Lab-2"

kali@kali: ~/Desktop/Language-Processors-Lab/Lab-2

File Actions Edit View Help

```
kali@kali:~/Desktop/Language-Processors-Lab/Lab-2$ gcc lex.yy.c
kali@kali:~/Desktop/Language-Processors-Lab/Lab-2$ ./a.out
int i = 0,a = 10;
        keywords : int  separator :     identifier : i  separator :     operator : =    separator :     integer : 0     separator : ,   identifier : a  separator :     op
erator : =      separator :     integer : 10    separator : ;
int i=0,a=10;
        keywords : int  separator :     identifier : i  operator : =    integer : 0     separator : ,   identifier : a  operator : =    integer : 10    separator : ;
int A[5];
        keywords : int  separator :     identifier : A  integer : 5     separator : ;
int sum(){}
        keywords : int  separator :     identifier : sum        separator : (   separator : )   separator : {   separator : }
^C
kali@kali:~/Desktop/Language-Processors-Lab/Lab-2$ ./a.out
int i=0,a=10;
        keywords : int  separator :     identifier : i  operator : =    integer : 0     separator : ,   identifier : a  operator : =    integer : 10    separator : ;
int A[5];
        keywords : int  separator :     identifier : A  integer : 5     separator : ;
int sum()
        keywords : int  separator :     identifier : sum        separator : (   separator : )

 total no. of token = 20
kali@kali:~/Desktop/Language-Processors-Lab/Lab-2$
::1             ff02::1         ff02::2         ip6-allnodes    ip6-allrouters  ip6-localhost   ip6-loopback    kali            localhost
kali@kali:~/Desktop/Language-Processors-Lab/Lab-2$ sS
```

Right Ctrl

G.Vikas
411924

Task – 2

```
#include <bits/stdc++.h>
using namespace std;

set<char> ss;
bool dfs(char i, char org, char last, map<char,vector<vector<char>>>
&mp){
   bool rtake = false;
for(auto r : mp[i]){
bool take = true;
for(auto s : r){
if(s == i) break;
        if(!take) break;
        if(!(s>='A'&&s<='Z')&&s!='e'){
           ss.insert(s);
break;
        }
        else if(s == 'e'){
if(org == i||i == last)
           ss.insert(s);
rtake = true;
           break;
```

```
                                                    }
else{
                                                        take = dfs(s,org,r[r.size()-1],mp);
rtake |= take;
        }
      }
   }
   return rtake;
}

int main(){
   int i,j;
   ifstream fin("input.txt");    string num;
vector<int> fs;    vector<vector<int>> a;
map<char,vector<vector<char>>> mp;
   char start;
   bool flag = 0;
   cout<<"Grammar: "<<'\n';
while(getline(fin,num)){       if(flag ==
0) start = num[0],flag = 1;
      cout<<num<<'\n';
vector<char> temp;       char s
= num[0];
for(i=3;i<num.size();i++){
if(num[i] == '|'){
mp[s].push_back(temp);
            temp.clear();
        }
        else temp.push_back(num[i]);
      }
      mp[s].push_back(temp);
   }
   map<char,set<char>> fmp;
for(auto q : mp){
ss.clear();
      dfs(q.first,q.first,q.first,mp);
```

```
    for(auto g : ss) fmp[q.first].insert(g);
  }

  cout<<'\n';
cout<<"FIRST: "<<'\n';
for(auto q : fmp){
    string ans = "";
ans += q.first;       ans

+= " = {";       for(char r :
q.second){
        ans += r;
        ans += ',';
    }

    ans.pop_back();
ans+="}";
    cout<<ans<<'\n';
  }

  map<char,set<char>> gmp;
gmp[start].insert('$');    int count =
10;    while(count--){       for(auto q
: mp){           for(auto r : q.second){
for(i=0;i<r.size()-1;i++){
if(r[i]>='A'&&r[i]<='Z'){
                if(!(r[i+1]>='A'&&r[i+1]<='Z')) gmp[r[i]].insert(r[i+1]);
else {
                                                    c
                                                    h
                                                    a
                                                    r
                                                    t
                                                    e
                                                    m
                                                    p
                                                    =
```

```
r
[
i
+
1
]
;
i
n
t
if(*fmp[temp].begin()=='e'){
for(auto g : fmp[temp]){
if(g=='e') continue;
j++;
if(j<r.size()){
j
=
i
+
1
;
while(temp>='A'&&temp<='Z')
{
gmp[r[i]].insert(g);
}
```

```
                                                    temp =
                                                    r[j];
                                                    if(!(tem
break;                                              p>='A'
                                                    &&temp
                                                    <='Z'))
                                                    {
                                                       gmp[r[i]].insert(temp);
                          }
                       }
                                                   else{
                                                      for(auto g : gmp[q.first])
break;                                                gmp[r[i]].insert(g);

                                                    }
}
else{
                                                    f
                                                       o
                                                       r
                                                       (
                                                       a
                                                       u
                                                       t
                                                       o
                                                       g
                                                       :
                                                       f
                                                       m
                                                       p
                                                       [
                                                       t
                                                       e
                                                       m
                                                       p
                                                       ]
                                                       )
```

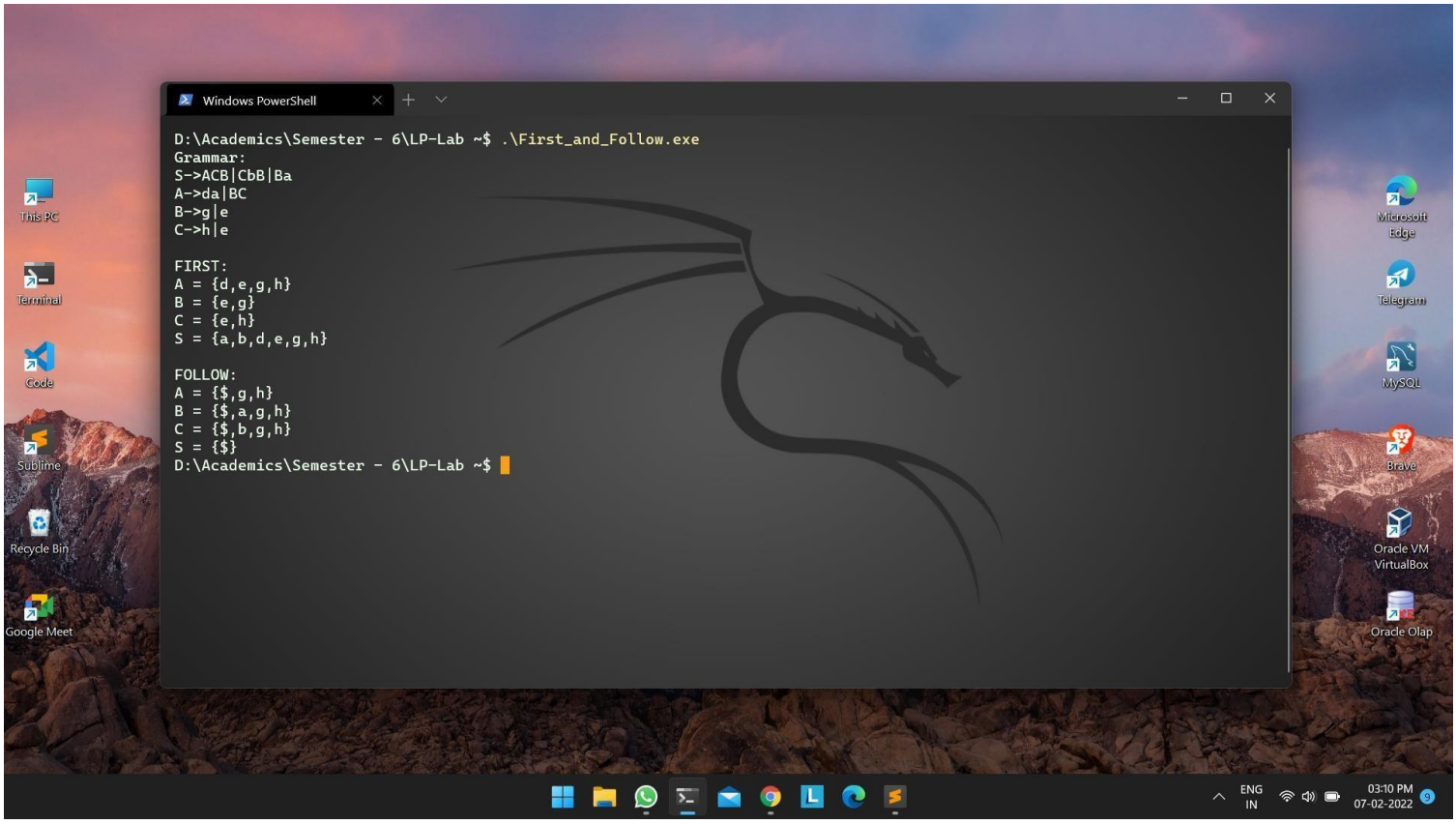
```
{                                                    i
g                                                    n
m                                                    s
p                                                    e
[                                                    r
r                                                    t
[                                                    (
i                                                    g
]                                                    )
]                                                    ;
.
                        }
                        break;
                    }
                }
            }
        }
        }
        if(r[r.size()-1]>='A'&&r[r.size()-1]<='Z'){
for(auto g : gmp[q.first]) gmp[r[i]].insert(g);
        }
    }
  }
}

cout<<'\n';
cout<<"FOLLOW: "<<'\n';
for(auto q : gmp){
string ans = "";        ans
+= q.first;        ans += "
= {";        for(char r :
q.second){
        ans += r;
```

```
        ans += ',';
    }
    ans.pop_back();
ans+="}";
    cout<<ans<<'\n';
  }
  return 0;
```

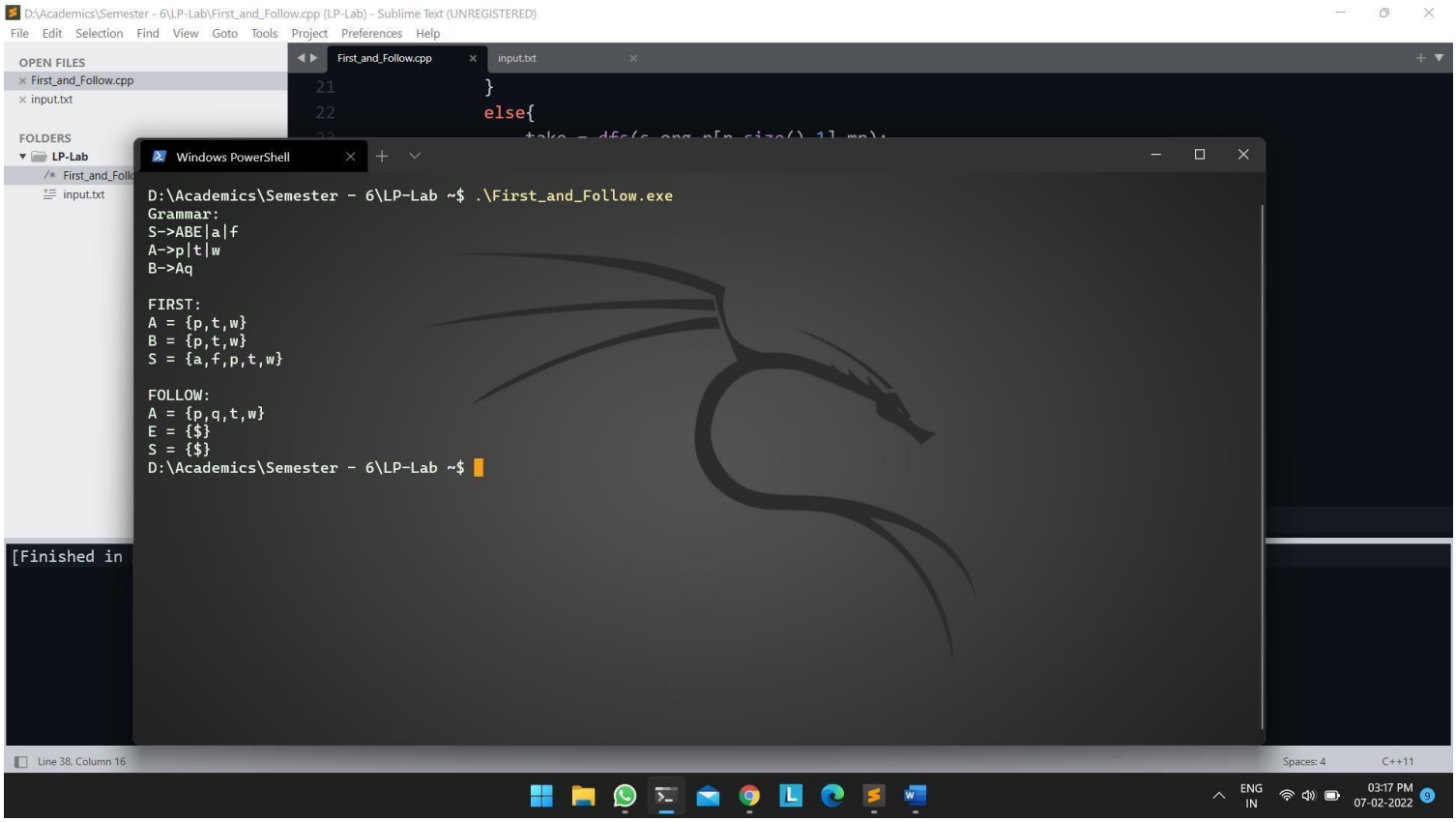
G.Vikas

411924

Grammar – 1:



Grammar – 2:


D:\Academics\Semester - 6\LP-Lab\First_and_Follow.cpp (LP-Lab) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
OPEN FILES
First_and_Follow.cpp
input.txt
FOLDERS
LP-Lab
First_and_Follo
input.txt
First_and_Follow.cpp
input.txt
}
else{
Windows PowerShell
D:\Academics\Semester - 6\LP-Lab ~$.\First_and_Follow.exe
Grammar:
S->ABE|a|f
A->p|t|w
B->Aq
FIRST:
A = {p,t,w}
B = {p,t,w}
S = {a,f,p,t,w}
FOLLOW:
A = {p,q,t,w}
E = {$}
S = {$}
D:\Academics\Semester - 6\LP-Lab ~$
[Finished in
Line 38, Column 16
Spaces: 4
C++11
ENG
IN
03:17 PM
07-02-2022

G.Vikas

411924

Task – 3

Q-1:

```
%{
 #include<stdio.h>
 #include<ctype.h>
 #include<stdlib.h>
%}


%token id num
%left '+' '-'
%left '*' '/'
%%
S:S'\n'{printf("The expression is valid"); exit(0);}
|error'\n'{printf("\nThe expression is not valid"); exit(0);}
|E
;
E:E'+'E
|E'-'E
|E'*'E
|E'/'E
|'('E')'
|id
```

```
|num
;
%%

main()
```

```
{
printf("Enter the expression:");
yyparse();        //parsing
 }

 yylex()
 {
 char c;
 while((c=getchar())==' ');
 if(isdigit(c))
return num;

if(isalpha(c))
return id;
return c;
 }
 yyerror(char *s)
 {
 printf("%s",s);
 }
```

Q-2:

Lex-file:

```
DIGIT [0-9]+\.?|[0-9]*\.[0-9]+
%%



[ ]



{DIGIT}    {yylval=atof(yytext);return NUM;}

\n|.            {return yytext[0];}
```

Yacc File:

```
%{
    #include<ctype.h>
    #include<stdio.h>
    #define YYSTYPE double
```

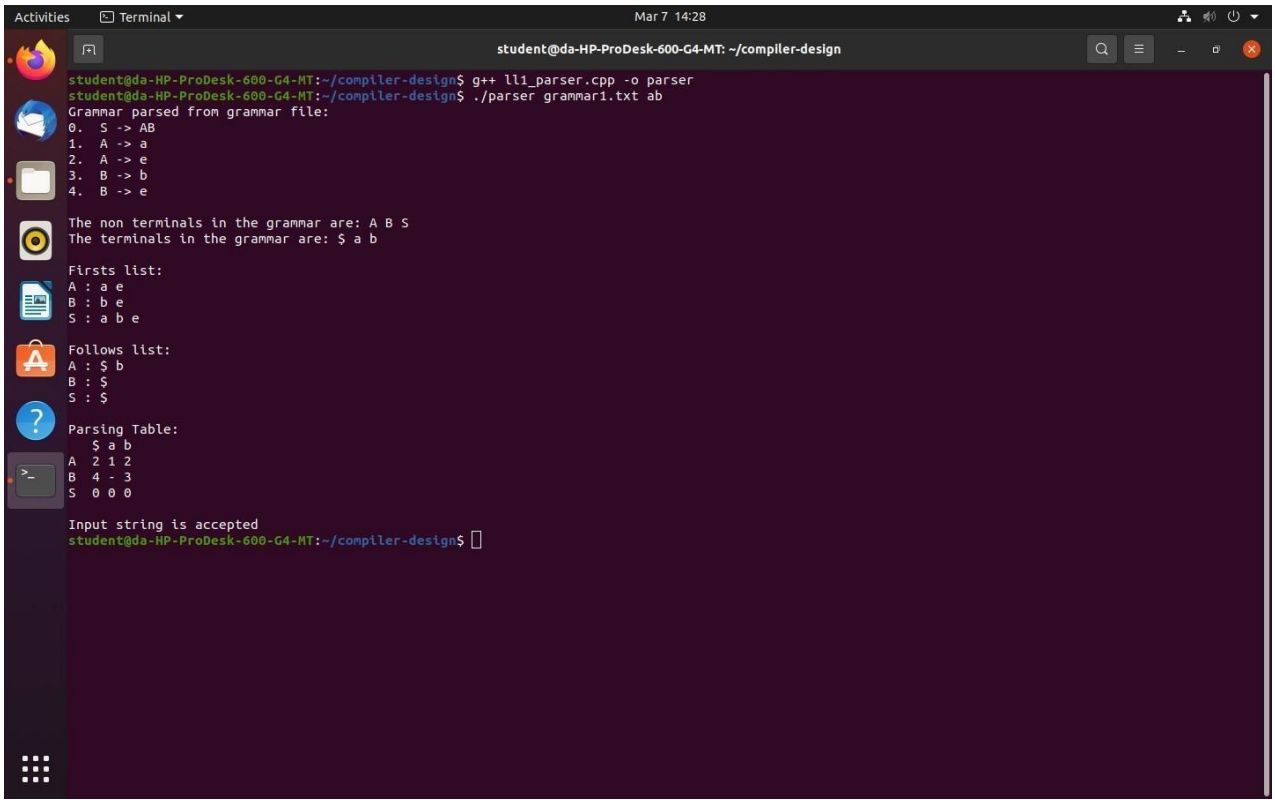
```
%}

%token NUM

%left '+' '-'
%left '*' '/'
%right UMINUS

%%

S       : S E '\n' { printf("Answer: %g \nEnter:\n", $2); }
        | S '\n'
        |
        | error '\n' { yyerror("Error: Enter once more...\n" );yyerrok; }
        ;
E       : E '+' E    { $$ = $1 + $3; }
        | E'-'E    { $$=$1-$3; }
        | E'*'E    { $$=$1*$3; }
        | E'/'E    { $$=$1/$3; }
        | '('E')'    { $$=$2; }
        | '-'E %prec UMINUS { $$= -$2; }
        | NUM
        ;

%%
#include "lex.yy.c"
int main()
{
   printf("Enter the expression: ");
yyparse();
}
```

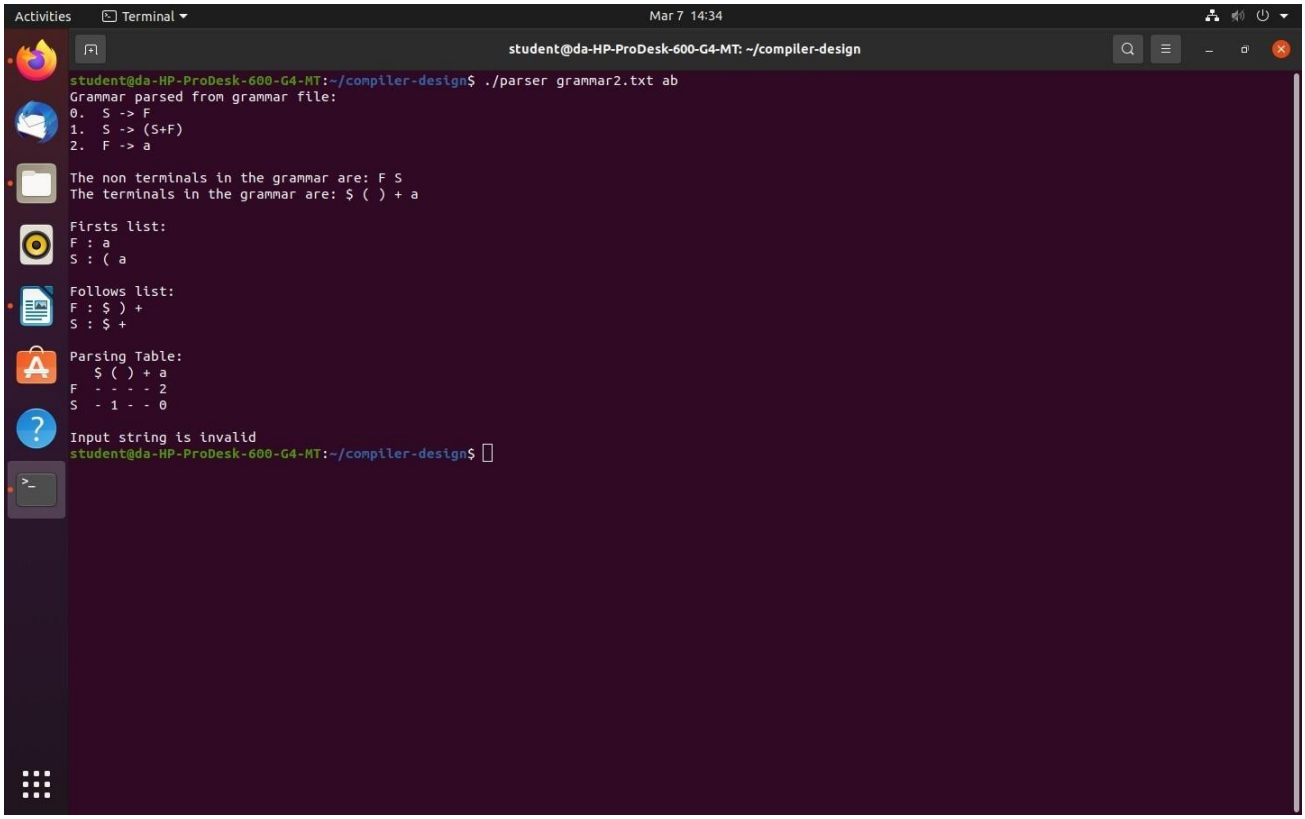
G.Vikas
411924

Task – 4

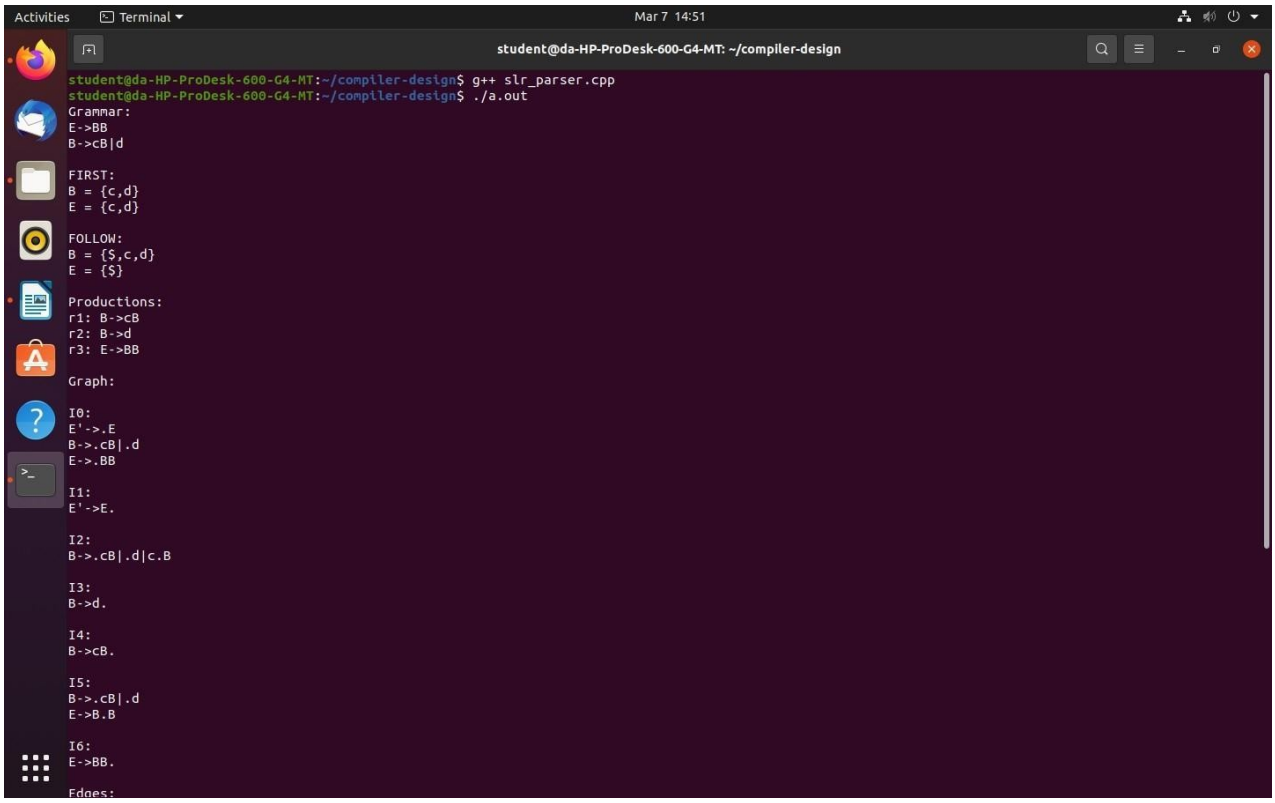
Implementing LL(1)-Parser and SLR(1)-Parser:

LL(1) – Parser:




Activities
Terminal
Mar 7 14:34
student@da-HP-ProDesk-600-G4-MT: ~/compiler-design
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$ ./parser grammar2.txt ab
Grammar parsed from grammar file:
0. S -> F
1. S -> (S+F)
2. F -> a
The non terminals in the grammar are: F S
The terminals in the grammar are: $ ( ) + a
Firsts list:
F : a
S : (a
Follows list:
F : $) +
S : $ +
Parsing Table:
$ ( ) + a
F - - - - 2
S - 1 - - 0
Input string is invalid
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$

SLR(1) – Parser :


Activities
Terminal
Mar 7 14:51
student@da-HP-ProDesk-600-G4-MT: ~/compiler-design
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$ g++ slr_parser.cpp
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$ ./a.out
Grammar:
E->BB
B->cB|d
FIRST:
B = {c,d}
E = {c,d}
FOLLOW:
B = {$,c,d}
E = {$}
Productions:
r1: B->cB
r2: B->d
r3: E->BB
Graph:
I0:
E'->.E
B->.cB|.d
E->.BB
I1:
E'->E.
I2:
B->.cB|.d|c.B
I3:
B->d.
I4:
B->cB.
I5:
B->.cB|.d
E->B.B
I6:
E->BB.
Edges:

```
B->.cB|.d
E->.BB

I1:
E'->E.

I2:
B->.cB|.d|c.B

I3:
B->d.

I4:
B->cB.

I5:
B->.cB|.d
E->B.B

I6:
E->BB.

Edges:
I0 -> E -> I1
I0 -> c -> I2
I0 -> d -> I3
I0 -> B -> I5
I2 -> c -> I2
I2 -> d -> I3
I2 -> B -> I4
I5 -> c -> I2
I5 -> d -> I3
I5 -> B -> I6

Parsing Table:
St.             Action & Goto
        $       c       d       B       E
I0      -       S2      S3      5       1
I1      AC      -       -       -       -
I2      -       S2      S3      4       -
I3      r2      r2      r2      -       -
I4      r1      r1      r1      -       -
I5      -       S2      S3      6       -
I6      r3      -       -
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$
```

```
[1]+  Done                    gedit inputslr.txt
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$ ./a.out
Grammar:
E->E+T
E->T
T->T*F
T->F
F->(E)
F->@

FIRST:
E = {(,@}
F = {(,@}
T = {(,@}

FOLLOW:
E = {$,),+}
F = {$,),*,+}
T = {$,),*,+}

Productions:
r1: E->E+T
r2: E->T
r3: F->(E)
r4: F->@
r5: T->T*F
r6: T->F

Graph:

I0:
E'->.E
E->.E+T|.T
F->.(E)|.@
T->.F|.T*F

I1:
E'->E.
E->E.+T

I2:
E->E+.T
F->.(E)|.@
T->.F|.T*F

I3:
```

```
I3:
E->E+T.
T->T.*F

I4:
E->.E+T|.T
F->.(E)|.@|(.E)
T->.F|.T*F

I5:
E->E.+T
F->(E.)

I6:
E->T.
T->T.*F

I7:
F->@.

I8:
F->(E).

I9:
T->F.

I10:
F->.(E)|.@
T->T*.F

I11:
T->T*F.

I12:
E->T.

Edges:
I0 -> E -> I1
I0 -> T -> I12
I0 -> ( -> I4
I0 -> @ -> I7
I0 -> F -> I9
I0 -> T -> I6
I1 -> + -> I2
I2 -> T -> I3
I2 -> ( -> I4
```

```
E->T.

Edges:
I0 -> E -> I1
I0 -> T -> I12
I0 -> ( -> I4
I0 -> @ -> I7
I0 -> F -> I9
I0 -> T -> I6
I1 -> + -> I2
I2 -> T -> I3
I2 -> ( -> I4
I2 -> @ -> I7
I2 -> F -> I9
I4 -> E -> I5
I4 -> T -> I6
I4 -> ( -> I4
I4 -> @ -> I7
I4 -> F -> I9
I5 -> + -> I2
I7 -> + -> I2
I7 -> ) -> I8
I9 -> * -> I10
I10 -> ( -> I4
I10 -> @ -> I7
I10 -> F -> I11
I11 -> * -> I10

Parsing Table:
St.             Action & Goto
        $       (       )       *       +       @       E       F       T
I0      -       S4      -       -       -       S7      1       9       12
I1      -       -       -       -       S2      -       -       -       -
I2      -       S4      -       -       -       S7      -       9       3
I3      r1      r0      r1      r0      r0      r1      r0      -       -       -
I4      -       S4      -       -       -       S7      5       9       6
I5      -       -       -       -       S2      -       -       -       -
I6      r2      r0      r2      r0      r0      r2      r0      -       -       -
I7      -       -       S8      -       S2      -       -       -       -
I8      r3      r3      r3      r3      -       -       -
I9      -       -       -       S10     -       -       -       -       -
I10     -       S4      -       -       -       S7      -       11      -
I11     -       -       -       S10     -       -       -       -       -
I12     r2      r2      r2      -       -       -
student@da-HP-ProDesk-600-G4-MT:~/compiler-design$
```

G.Vikas
411924

Task – 5

Q-1:
Lex – File:

```
%{
/* Definition section */
#include<stdio.h> #include
"y.tab.h"
extern int yylval;
%}

/* Rule Section */
%%
[0-9]+ {
            yylval=atoi(yytext);
            return NUMBER;

      }
[\t] ;

[\n] return 0;

. return yytext[0];
```

%%

```
int yywrap()
{
return 1;
}
```

Yacc – File:

```
%{
/* Definition section */ #include<stdio.h>
int flag=0;
%}

%token NUMBER

%left '+' '-'

%left '*' '/' '%'

%left '(' ')'

/* Rule Section */
%%

ArithmeticExpression: E{

        printf("\nResult=%d\n", $$);

        return 0;

        };

E:E'+'E {$$=$1+$3;}

|E'-'E {$$=$1-$3;}
```

```
|E'*'E {$$=$1*$3;}

|E'/'E {$$=$1/$3;}

|E'%'E {$$=$1%$3;}
|'('E')' {$$=$2;}

| NUMBER {$$=$1;}

;

%%

//driver code
void main()
{
printf("\nEnter Any Arithmetic Expression which
                    can have operations Addition,
                    Subtraction, Multiplication, Division,
                              Modulus and Round brackets:\n");

yyparse(); if(flag==0)
printf("\nEntered arithmetic expression is Valid\n\n");
}

void yyerror()
{
printf("\nEntered arithmetic expression is Invalid\n\n"); flag=1;
}
```

Q-2:

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
```

```
void followfirst(char , int , int);
void findfirst(char , int , int); void
follow(char c);

int count,n=0;
char calc_first[10][100]; char
calc_follow[10][100]; int
m=0;
char production[10][10], first[10];
char f[10]; int k; char ck;
int e;

int main(int argc,char **argv)
{
	int jm=0;
int km=0;
	int i,choice;
char c,ch;
	printf("How many productions ? :");
	scanf("%d",&count);
	printf("\nEnter %d productions in form A=B where A and B are
grammar symbols :\n\n",count);	for(i=0;i<count;i++)
	{
		scanf("%s%c",production[i],&ch);
	}
	int kay;	char done[count];
int ptr = -1;	for(k=0;k<count;k++){
	for(kay=0;kay<100;kay++){
			calc_first[k][kay] = '!';
		}
	}
	int point1 = 0,point2,xxx; for(k=0;k<count;k++)
	{
```

```
            c=production[k][0];
            point2 = 0;     xxx =
      0;
            for(kay = 0; kay <= ptr; kay++)
                  if(c == done[kay])
                        xxx = 1;

            if (xxx == 1)
                  continue;
            findfirst(c,0,0);
            ptr+=1;
            done[ptr] = c;                  printf("\n

First(%c)= { ",c);
calc_first[point1][point2++] = c;
for(i=0+jm;i<n;i++){                        int lark =
0,chk = 0;
for(lark=0;lark<point2;lark++){
                        if (first[i] == calc_first[point1][lark]){
                              chk = 1;
                  break;

                        }
                  }
                  if(chk == 0){
                        printf("%c, ",first[i]);
                        calc_first[point1][point2++] = first[i];
                  }

            }
            printf("}\n");
            jm=n;
point1++;

      }
      printf("\n");
      printf("                                                \n\n");
```

```
        char donee[count];        ptr = -1;
for(k=0;k<count;k++){
for(kay=0;kay<100;kay++){
```

```
                calc_follow[k][kay] = '!';
            }
        }
        point1 = 0;
        int land = 0; for(e=0;e<count;e++)
            {               ck=production[e][0];
                point2 = 0;
            xxx = 0;
            for(kay = 0; kay <= ptr; kay++)
                if(ck == donee[kay])
                    xxx = 1;
            if (xxx == 1)
                continue;

land += 1;
follow(ck);             ptr+=1;
            donee[ptr] = ck;
            printf(" Follow(%c) = { ",ck);
calc_follow[point1][point2++] = ck;
for(i=0+km;i<m;i++){                    int lark =
0,chk = 0;
for(lark=0;lark<point2;lark++){
                    if (f[i] == calc_follow[point1][lark]){
                    chk = 1;
                        break;
                    }
                }
                if(chk == 0){
                    printf("%c, ",f[i]);
                    calc_follow[point1][point2++] = f[i];
                }

            }
            printf(" }\n\n");
            km=m;
            point1++;
```

}

```
        char ter[10]; for(k=0;k<10;k++){
                ter[k] = '!';
        }
        int ap,vp,sid = 0; for(k=0;k<count;k++){
        for(kay=0;kay<count;kay++){
                            if(!isupper(production[k][kay]) && production[k][kay]!=
'#' && production[k][kay] != '=' && production[k][kay] != '\0'){
                                    vp = 0;
                                    for(ap = 0;ap < sid; ap++){
if(production[k][kay] == ter[ap]){
                                                    vp = 1;
                            break;
                                            }
                                    }
                                    if(vp == 0){
                                            ter[sid] = production[k][kay];
                                            sid ++;
                                    }

                            }
                    }
        }
        ter[sid] = '$';
        sid++;
        printf("\n\t\t\t\t\t\t\t The LL(1) Parsing Table for the above
grammer :-");
          printf("\n\t\t\t\t\t\t\t^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
^^^^^^^^^^^^^^^^^\n");
        printf("\n\t\t\t=============================
=========================================

=========================================
===\n");
        printf("\t\t\t\t|\t");
for(ap = 0;ap < sid; ap++){
                printf("%c\t\t",ter[ap]);
```

}

```
	printf("\n\t\t\t==========================================================================================================\n");
	char first_prod[count][sid];
	for(ap=0;ap<count;ap++){
		int destiny = 0;
		k = 2;
		int ct = 0; char
		tem[100];
		while(production[ap][k] != '\0'){
			if(!isupper(production[ap][k])){
				tem[ct++] = production[ap][k];
				tem[ct++] = '_';
				tem[ct++] = '\0';
				k++;
		break;

			}
			else{
	int zap=0;
int tuna = 0;

				for(zap=0;zap<count;zap++){
					if(calc_first[zap][0] == production[ap][k]){
				for(tuna=1;tuna<100;tuna++){

		if(calc_first[zap][tuna] != '!'){
								tem[ct++] = calc_first[zap][tuna];
							}
							else
						break;
			}
					break;
					}
				}
				tem[ct++] = '_';
```

}

```
                k++;
            }
            int zap = 0,tuna;
for(tuna = 0;tuna<ct;tuna++){
                if(tem[tuna] == '#'){
                    zap = 1;
                }
                else if(tem[tuna] == '_'){
                    if(zap == 1){
                    zap = 0;
                    }
                    else
                    break;
                }
                else{
                    first_prod[ap][destiny++] = tem[tuna];
                }
            }
        }
        char table[land][sid+1];
ptr = -1;
        for(ap = 0; ap < land ; ap++){
for(kay = 0; kay < (sid + 1) ; kay++){
                table[ap][kay] = '!';
            }
        }
        for(ap = 0; ap < count ; ap++){
            ck = production[ap][0];
            xxx = 0;
            for(kay = 0; kay <= ptr; kay++)
                if(ck == table[kay][0])
                    xxx = 1;
            if (xxx == 1)
```

```
                continue;
        else{                   ptr
= ptr + 1;
                table[ptr][0] = ck;
            }
        }
        for(ap = 0; ap < count ; ap++){
            int tuna = 0;
            while(first_prod[ap][tuna] != '\0'){
        int to,ni=0;            for(to=0;to<sid;to++){
            if(first_prod[ap][tuna] == ter[to]){
        ni = 1;
                        }
                }
                if(ni == 1){
                        char xz = production[ap][0];
                        int cz=0;
                        while(table[cz][0] != xz){
                        cz = cz + 1;
                        }
                        int vz=0;
                        while(ter[vz] != first_prod[ap][tuna]){
                                vz = vz + 1;
                        }

                        table[cz][vz+1] = (char)(ap + 65);
                }
                tuna++;
            }
        }
        for(k=0;k<sid;k++){
for(kay=0;kay<100;kay++){
                if(calc_first[k][kay] == '!'){
            break;
                }
                else if(calc_first[k][kay] == '#'){
```

```
                int fz = 1;
                while(calc_follow[k][fz] != '!'){
                        char xz = production[k][0];
                        int cz=0;
                        while(table[cz][0] != xz){
                cz = cz + 1;

                        }
                        int vz=0;
                        while(ter[vz] != calc_follow[k][fz]){
                                vz = vz + 1;
                        }
                        table[k][vz+1] = '#';
                        fz++;
                }
                break;
            }
        }
    }
    for(ap = 0; ap < land ; ap++){
        printf("\t\t\t   %c\t|\t",table[ap][0]);
        for(kay = 1; kay < (sid + 1) ; kay++){
            if(table[ap][kay] == '!')
        printf("\t\t");
            else if(table[ap][kay] == '#')
                printf("%c=#\t\t",table[ap][0]);
            else{

                int mum = (int)(table[ap][kay]);
        mum -= 65;
                printf("%s\t\t",production[mum]);
            }
        }
        printf("\n");
        printf("\t\t\t-----------------------------------------------------------------
-------------------------------------------------------");
```

```
printf("\n");
```

```
        }
        int j;
        printf("\n\nPlease enter the desired INPUT STRING = ");
char input[100];    scanf("%s%c",input,&ch);
        printf("\n\t\t\t\t\t==============================
=========================================
=====\n");
        printf("\t\t\t\t\t\tStack\t\t\tInput\t\t\tAction");
        printf("\n\t\t\t\t\t==============================
=========================================

=====\n");
        int i_ptr = 0,s_ptr = 1;
        char stack[100];
        stack[0] = '$'; stack[1] =
        table[0][0]; while(s_ptr !=

        -1){ printf("\t\t\t\t\t\t");
                int vamp = 0;
                for(vamp=0;vamp<=s_ptr;vamp++){
                        printf("%c",stack[vamp]);
                }

                printf("\t\t\t");   vamp   =
        i_ptr;    while(input[vamp]    !=
        '\0'){
                        printf("%c",input[vamp]);
                        vamp++;
                }
                printf("\t\t\t");
char her = input[i_ptr];
                char him = stack[s_ptr];
        s_ptr--;
if(!isupper(him)){
if(her == him){
i_ptr++;
                                printf("POP ACTION\n");
```

}

```
                else{
                        printf("\nString Not Accepted by LL(1) Parser !!\n");
                exit(0);
                }
        }
        else{
                for(i=0;i<sid;i++){
    if(ter[i] == her)
                        break;
                }
                char produ[100];
for(j=0;j<land;j++){                    if(him ==
table[j][0]){                           if (table[j][i+1] == '#'){
                        printf("%c=#\n",table[j][0]);
        produ[0] = '#';
                                produ[1] = '\0';
                        }
                        else if(table[j][i+1] != '!'){
                                int mum = (int)(table[j][i+1]);
                                mum -= 65;
                                strcpy(produ,production[mum]);
                                printf("%s\n",produ);
                                                }

                                                else{
                                                        printf("\nString Not
Parser !!\n");                                          Accepted by LL(1)

                                                        exit(0);
                        }
                }
                }
                int le = strlen(produ);
        le = le - 1;
if(le == 0){
                        continue;
```

}

```
                for(j=le;j>=2;j--){
                        s_ptr++;
                        stack[s_ptr] = produ[j];
                }
        }
    }
    printf("\n\t\t\t==================================
=========================================================
=========================================================
=====\n");
    if (input[i_ptr] == '\0'){
        printf("\t\t\t\t\t\t\t\tYOUR STRING HAS BEEN ACCEPTED
                                !!\n");
                                        }
    else
                                        printf("\n\t\t\t\t\t\t\t\tYOUR
                                !!\n");  STRING HAS BEEN REJECTED

    printf("\t\t\t==================================
=========================================================
=========================================================
====\n");
}
void follow(char c)
{
    int i ,j;
    if(production[0][0]==c){
f[m++]='$';

    }
    for(i=0;i<10;i++)
    {
        for(j=2;j<10;j++)
        {
            if(production[i][j]==c)
            {
```

```
if(production[i][j+1]!='\0'){
```

```
                followfirst(production[i][j+1],i,(j+2));
            }
        if(production[i][j+1]=='\0'&&c!=production[i][0]){
            follow(production[i][0]);
            }

        }
    }
  }
}

void findfirst(char c ,int q1 , int q2)
{
    int j;
if(!(isupper(c))){
        first[n++]=c;
    }
    for(j=0;j<count;j++)
    {
        if(production[j][0]==c)
        {
            if(production[j][2]=='#'){
if(production[q1][q2] == '\0')
first[n++]='#';
                                        else if(production[q1][q2]
0))                                     != '\0' && (q1 != 0 || q2 !=

                                       {
                                            findfirst(production[q1]
                                            [q2], q1, (q2+1));
                                       }
                                  else
                                        first[n++]='#';
            }

            else if(!isupper(production[j][2])){
```

```
        first[n++]=production[j][2];
}
else {
```

```
                        findfirst(production[j][2], j, 3);
                }
            }
        }
}

void followfirst(char c, int c1 , int c2)
{    int
k;
   if(!(isupper(c)))
            f[m++]=c;
else{           int
i=0,j=1;
            for(i=0;i<count;i++)
            {
                if(calc_first[i][0] == c)
            break;
            }
            while(calc_first[i][j] != '!')
            {
                if(calc_first[i][j] != '#'){
                        f[m++] = calc_first[i][j];
                }
                else{
                        if(production[c1][c2] == '\0'){
                                follow(production[c1][0]);
                        }

                        else{
                                followfirst(production[c1][c2],c1,c2+1);
                        }
            }
                j++;
            }
        }
}
```