ASSIGNMENT 02

Left Recursion Removal-

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

int main(){
	printf("Enter no. of productions.");
	int i,n;
	scanf("%d",&n);
	printf("Note:\nEnter valid Productions\n");
	for(i=0;i<n;i++){
		char p[20];
		printf("\nEnter Production %d:",i+1);
		scanf("%s",p);
		//printf("%s",p);

		int len=strlen(p);
		//printf("%d",len);
		int j,pipe_index;
		if(p[0]==p[3] && p[1]=='-' && p[2]=='>'){
			for(j=0;j<len;j++){
				if(p[j]=='|')	{pipe_index=j;		break;}
				else				pipe_index=21;
			}
			//printf("%d ",pipe_index);
			int  rest=pipe_index-3-1;
			//printf("%d ",rest);
			int position=4;
			int length=rest,c=0;
			char sub1[20],sub2[20],sub3[20];
			while (c < length) {
			sub1[c] = p[position+c];
			c++;
			}
			sub1[c] = '\0';
			//printf("Required substring is \"%s\"\n", sub1);

			position=0;
			length=3;
			c=0;
			while (c < length) {
			sub2[c] = p[position+c];
			c++;
			}
			sub2[c] = '\0';

			sub3[0]=p[pipe_index+1];
			sub3[1] = '\0';

			//printf("Required substring is \"%s\"\n", sub3);
```

```
                        char p1[20],p2[20];
                        p1[0]= '\0';
                        strcat(p1,sub2);
                        strcat(p1,sub3);
                        p2[0]=p[0];
                        p2[1]='\'';
                        strcat(p1,p2);

                        printf("%s\n", p1);

                        p2[2]='-';
                        p2[3]='>';
                        p2[4]= '\0';
                        strcat(p2,sub1);

                        char e_dash[3];
                        e_dash[0]=p[0];
                        e_dash[1]='\'';
                        e_dash[2]='\0';
                        strcat(p2,e_dash);
                        //printf("Required substring is \"%s\"\n", p2);

                        char epsilon[3];
                        epsilon[0]='|';
                        epsilon[1]='@';
                        epsilon[2]='\0';

                        if(sub3[0]!='\0' && pipe_index!=21){
                                strcat(p2,epsilon);
                        }
                        printf("%s\n", p2);
                }
                else    printf("\nNot a Left-Recursive Grammar");
        }
        printf("\n\nNote:\nEpsilon symbol='@'");
}
```

Output-

```
Enter no. of productions: 1
Note:
Enter valid Productions

Enter Production 1:
E->E+T|T

E->TE'
E'->+TE'|@
```

ASSIGNMENT 03

First and Follow-

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
void FIRST(char[],char );
void addToResultSet(char[],char);
int size,m=0;
char a[10][10],followResult[10];
void follow(char c);
void temp(char c);
void addToResultFollow(char);

void main(){
   int i;
   char c;
   char result[20];
   printf("How many number of productions ? :");
   scanf(" %d",&size);
   printf("Epsilon==>@\n");
   for(i=0;i<size;i++){
      printf("Enter productions Number %d : ",i+1);
      scanf(" %s",a[i]);
   }
   int flag=0;
   for(i=0;i<size;i++){
         if( islower(a[i][0]))    flag=1;
                  if(a[i][0]==a[i][3])     flag=2;
                  if(flag==1 || flag==2) break;
         }

         if(flag==0){

   int b[size],cx=0,cnt=0;
   for(i=0;i<size;i++) b[i]='\0';                          //initialize array b[] by NULL
   for(i=0; i<size; i++){                                           //stores LHS values in b[] array
         cx=0;
         int j;
   for(j=0;j<i+1;j++){                                                                  //check duplicates
      if(a[i][0] == b[j]){
          cx=1;
          break;
          }
      }
   if(cx !=1){                                                                                   //store
unique values in b[]
     b[cnt] = a[i][0];
     cnt++;
   }
   }
```

```
    //for(i=0;i<cnt;i++)             printf("%c ",b[i]);

    int ii=0;
    while(ii!=cnt){
            c=b[ii];
        FIRST(result,c);                                                              //Compute FIRST();
        printf("\n FIRST(%c)= { ",c);
        for(i=0;result[i]!='\0';i++)
        printf(" %c ",result[i]);                                          //Display result
        printf("}\n");
        ii++;
    }
    printf("\n");
    printf("\n");


    int ij=0;
    while(ij!=cnt){
            c=b[ij];            //printf("%c ",b[ij]);
        m=0;
        follow(c);                                                                                      //Compute
FOLLOW();
                        printf("\n FOLLOW(%c) = { ",c);
                        for(i=0;i<m;i++)
                                printf(" %c ",followResult[i]);                                 //Display result
                        printf(" }\n");
        ij++;
    }
            }
            else if(flag==1){
                        printf("\nNot a valid grammar.....\n");
            }
            else if(flag==2){
                        printf("\nGrammar has left recursion....\n");
            }
}


void follow(char c){
            int j,i;
    if(a[0][0]==c)                                              addToResultFollow('$');
 for(i=0;i<size;i++){
  for(j=3;j<strlen(a[i]);j++){
   if(a[i][j]==c){
    if(a[i][j+1]!='\0')                                   temp(a[i][j+1]);
    if(a[i][j+1]=='\0'&&c!=a[i][0])     follow(a[i][0]);
   }
  }
 }
}
```

```
void temp(char c){
      int k;
      if(!(isupper(c)))                addToResultFollow(c);
      for(k=0;k<size;k++){
            if(a[k][0]==c){
                        if(a[k][3]=='@')                               follow(a[k][0]);
            else if(islower(a[k][3]))   addToResultFollow(a[k][3]);
            else                                                       temp(a[k][3]);
          }
      }
}

void addToResultFollow(char c){
    int i;
    for( i=0;i<=m;i++){
            if(followResult[i]==c)          return;
            }
    followResult[m++]=c;
}

void FIRST(char* Result,char c){
    int i,j,k;
    char subResult[20];
    int epsilon;
    subResult[0]='\0';
    Result[0]='\0';
    if(!(isupper(c))){
        addToResultSet(Result,c);
        return ;
    }
    for(i=0;i<size;i++){
        if(a[i][0]==c){
                                    if(a[i][3]=='@')                                                              addToRe-
sultSet(Result,'@');
                        else{
                j=3;
                while(a[i][j]!='\0'){
                epsilon=0;
                FIRST(subResult,a[i][j]);
                for(k=0;subResult[k]!='\0';k++)         addToResultSet(Result,subResult[k]);
                 for(k=0;subResult[k]!='\0';k++){                        //{}
                     if(subResult[k]=='@'){
                         epsilon=1;
                         break;
                     }
                 }
                 if(!epsilon)                                break;
                 j++;
                }
            }
    }
}
return;
}

void addToResultSet(char Result[],char val){
   int k;
   for(k=0 ;Result[k]!='\0';k++){                              //{}
      if(Result[k]==val)                                                                  return;
      }
   Result[k]=val;
   Result[k+1]='\0';
}
```

Output-

```
How many number of productions ? :8
Epsilon==>@
Enter productions Number 1 : E->TX
Enter productions Number 2 : X->+TX
Enter productions Number 3 : X->@
Enter productions Number 4 : T->FY
Enter productions Number 5 : Y->*FY
Enter productions Number 6 : Y->@
Enter productions Number 7 : F->a
Enter productions Number 8 : F->(E)

 FIRST(E)= {  a  ( }

 FIRST(X)= {  +  @ }

 FIRST(T)= {  a  ( }

 FIRST(Y)= {  *  @ }

 FIRST(F)= {  a  ( }



 FOLLOW(E) = {  $  )  }

 FOLLOW(X) = {  $  )  }

 FOLLOW(T) = {  +  $  )  }

 FOLLOW(Y) = {  $  )  +  }

 FOLLOW(F) = {  *  +  $  )  }
```

ASSIGNMENT 04

LEXX – YACC Programs.
2. Count number of lines, character and words in a file.

```
%{
int charcount=0,wordcount=0,linecount=0;
%}
%%
[\n] {linecount++;wordcount++;charcount++;}
[\t] {charcount++,wordcount++;}
[" "] {charcount++,wordcount++;}
[^\n\t] {charcount++;}
%%
int main(){

        FILE *fp;
        char file[10];
        printf("Enter the filename");
        scanf("%s",file);
        fp=fopen(file,"r");
        yyin=fp;
        yylex();

        printf("No. of Characters = %d, No. of Words: %d, No. of
Lines: %d\n",charcount,wordcount,linecount);
return 0;
}
```

Testfile.txt-

```
hello world
hello world
hello
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question2$ ./a.out
Enter the filename file
No. of Characters = 30, No. of Words: 5, No. of Lines: 3
```

3. Remove comments from the C-code.

```
%{
extern char* yytext;
%}
%s COMMENT
%%
"/*" {BEGIN COMMENT;}
<COMMENT>"*/" {BEGIN 0;}
<COMMENT>\n      {;}
\/\/.*      {;}
<COMMENT>. {;}
.|\n      { fprintf(yyout,"%s",yytext);}
%%
int main(void){
        char file1[10],file2[10];
        printf("Enter the filename:\n");
        scanf("%s",file1);
        printf("Enter the output filename:\n");
        scanf("%s",file2);
        FILE *f1,*f2;
        f1=fopen(file1,"r");
        f2=fopen(file2,"w");
        yyin=f1;
        yyout=f2;
        yylex();
}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question3$ ./a.out
Enter the filename:
b.c
Enter the output filename:
a.txt
```

File- **b.c**

```
#include<stdio.h>
//one line comment
int main(){
int i,j;
/* multiple
line comment */
printf("%d",i);         }
```

File- **a.txt**

```
#include<stdio.h>

int main(){
int i,j;

printf("%d",i);         }
```

4. Check valid arithmetic expression.

```
%{
#include<stdio.h>
#include<string.h>

int operator_count=0,operand_count=0 l=0,j=0,top=-1,flag=1;
char operand[10][10],operator[10][10],a[100];
%}

%%
"{" {a[top++]='{';}
"[" {a[top++]='[';}
"(" {a[top++]='(';}

"}" { if(a[top]!='}' ){flag=0;return 0;}
                else top--;
        }
"]" { if(a[top]!=']' ){flag=0;return 0;}
                else top--;
        }
")" { if(a[top]!='(') {flag=0;return 0;}
                else top--;
        }

"+"|"-"|"*"|"/" {operator_count++; strcpy(operator[l++],yytext);}
[0-9]+|[a-zA-Z]|[a-zA-Z0-9_]* {operand_count++;strcpy(operand[j++],yytext);}

%%
int main(){
        printf("Enter the exprssion\n");
        yylex();
        if(flag==0)             printf("Invalid\n");

        //int i;
        else if(flag==1 && top==-1){

                if((operand_count-operator_count)==1){
                printf("Operand(s) is/are:\n");
                int i;
                for(i=0;i<j;++i)
                        printf("%s \n",operand[i]);
                printf("Operator(s) is/are:\n");
                for(i=0;i<l;++i)
                        printf("%s\n",operator[i]);
                }
                else            printf("Invalid\n");

        }
        else                    printf("Invalid\n");

}
```

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

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question4$ ./a.out
Enter the exprssion
a+b

Operand(s) is/are:
a
b
Operator(s) is/are:
+
```

5. Checking scentence to be simple or compound.

```
%{
int flag=1;
%}

%%
[ \t\n]+[aA][nN][dD][ \t\n]+ {flag=0;}
[ \t\n]+[oO][rR][ \t\n]+ {flag=0;}
[ \t\n]+[sS][iI][nN][cC][eE][ \t\n]+ {flag=0;}
[ \t\n]+[bB][eE][cC][aA][uU][sS][eE][ \t\n]+ {flag=0;}
[ \t\n]+[bB][uU][tT][ \t\n]+ {flag=0;}
.                    {;}
%%

int main(){
        printf("Enter scentence:\n");
        yylex();
        if(flag==1)
                printf("Sentence is simple\n");
        else
                printf("Sentence is compound\n");

}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question5$ ./a.out
Enter scentence:
System and Programming
Sentence is compound

abhishek@abhishek-HP-Notebook:~/Desktop/SP/question5$ ./a.out
Enter scentence:
System Programming
Sentence is simple
```

6. Check for whether string is keyword, identifier or not.

```
%{
#include <stdio.h>
int count =0;
%}

letter [a-zA-Z_]
digit [0-9]
id {letter}+|{letter}{digit}+
notid ({digit}|{letter})+
%%
(("int")|("float")|("char")|("case")|("default")|("if")|("for")|("printf")|("scanf")) {printf("%s is a
keyword\n", yytext);}
{id} {printf("%s is an identifier \n",yytext);count++;}
{notid} {printf("%s is not an identifier\n",yytext);}
. {;}
%%

int main(){
        FILE *fp;
        char file[10];
        printf("Enter the name of the file \n");
        scanf("%s",file);
        fp=fopen(file,"r");
        yyin=fp;
        yylex();
        printf("Total identifiers are : %d\n",count);
        return 0;
}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question6$ ./a.out
Enter the name of the file
Testfile.txt
int is a keyword
float is a keyword
12a is not an identifier
a12 is an identifier
abc is an identifier
Total identifiers are : 2
```

File- **Testfile.txt**

```
int
float
12a
a12
abc
```

7. Check for valid c-variables.

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

alphabet [a-zA-Z_]
number  [0-9]
%%
{alphabet}({alphabet}|{number})*({alphabet}|{number})* {flag=1;}
{number}|({number}|{alphabet})*                          {flag=0;}
%%

int main(){
	printf("Enter the variable \n");
	yylex();
	if(flag==1)		printf("Valid variable %s", yytext);
	else if(flag==0)	printf("Invalid variable %s",yytext);

}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question7$ ./a.out
Enter the variable
a12
Valid variable

abhishek@abhishek-HP-Notebook:~/Desktop/SP/question7$ ./a.out
Enter the variable
1ab
Invalid variable
```

8. Basic arithmetic operations, calculator.

yacc_q8.l

```
%{
#include<stdio.h>
#include<stdlib.h>
#include "y.tab.h"
extern int yylval;
%}

%%
[0-9]+ {
                yylval=atoi(yytext);
                return ID;
                }
[ \t]   ;
\n              return 0;
.               yyerror()
%%
```

yacc_q8.y

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

%token ID
%left '+' '-'
%left '*' '/'
%left '(' ')'
%%

expr : e{
                printf("result:%d\n",$$);
                return 0;
                }
e:      e'+'e {$$=$1+$3;}       |       e'-'e {$$=$1-$3;}       |       e'*'e {$$=$1*$3;}       |       e'/'e
{$$=$1/$3;}   |       '('e ')' {$$=$2;}         |       ID {$$=$1;}
        ;
%%
int main(){
        printf("enter the expression:\n");
        yyparse();
        }
yyerror(){
        printf("\n invalid expression\n");
        exit(0);
}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question8$ ./a.out
enter the expression:
2+(5*2)
result:12
```

9. Check string to be valid for grammar G: S→(a^n b^n), n>0.

yacc_q9.l

```
%{
#include "y.tab.h"
%}

%%
a {return A;}
b {return B;}
.|\n return yytext[0];
%%
```

yacc_q9.y

```
%{
#include<stdio.h>
void yyerror(char *s);
%}

%token A
%token B
%%

expr : S'\n' {printf("Accepted\n");exit(0);}
S : A S B        |          ;
%%

int main(){
	printf("Enter a string:\n");
	yyparse();
}

void yyerror(char *s){
	printf("Rejected\n");
	exit(1);
}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question9$ ./a.out
Enter a string:
aaaaabbb
Rejected

abhishek@abhishek-HP-Notebook:~/Desktop/SP/question9$ ./a.out
Enter a string:
aaabbb
Accepted
```

10. Check string to be valid for grammar G: S→$(a^n b)$, n>=10.

yacc_q10.l

```
%{      #include "y.tab.h"      %}

%%
[aA] {return A;}
[bB] {return B;}
\n {return E;}
. {return yytext[0];}
%%
```

yacc_q10.y

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

%token A B E

%%
stmt: A A A A A A A A A A S B E {printf("valid string\n");
            exit(0);}
;
S: S A
|
;
%%

int yyerror(char *msg)
{
printf("invalid string\n");
exit(0);
}

main(){
printf("enter the string\n");
yyparse();
}
```

Output-

```
abhishek@abhishek-HP-Notebook:~/Desktop/SP/question10$ ./a.out
enter the string
aaaab
invalid string

abhishek@abhishek-HP-Notebook:~/Desktop/SP/question10$ ./a.out
enter the string
aaaaaaaaaaaaaaaab
valid string
```