Q1) Program to count the number of vowels and consonants in a given string

```
%{
    #include<stdio.h>
    int  v=0,c=0;
%}
vowel [aeiouAEIOU]
rem [a-zA-Z]
%%
{vowel} {v++;}
{rem} {c++;}
. {}
%%
void main(){
    printf("Enter string:");
    yylex();
    printf("Number of vowels : %d\nNumber of consonants : %d\n",v,c);
}
int yywrap(){
    return 1;
}
```

Q2) Program to count the number of characters, words, spaces and lines in a given input file.

```
{
    #include<stdio.h>
    #include<stdlib.h>
    int c=0,w=0,s=0,l=1;
%}
WORD  [^ \n,\t\.:]+
BLANK [ ]
EOL [\n]
%%
{WORD} {w++;c+=yyleng;}
{BLANK} {s++;}
{EOL} {l++;}
. {}
%%
void main(int argc,char *argv[]){
    if(argc!=2)
    {
        printf("Invalid arguments");
        exit(0);
    }
    yyin=fopen(argv[1],"r");
    yylex();
    printf("S:%d\nC:%d\nW:%d\nNL:%d\n",s,c,w,l);
}
```

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

Q3) Program to count no of:

a) +ve and –ve integers

b) +ve and –ve fractions

```
%{
    #include<stdio.h>
    int pn=0,nn=0,fpn=0,fnn=0;
%}
%%
[+]?[0-9]+ {pn++;}
[-][0-9]+ {nn++;}
[+]?[0-9]*"."[0-9]+ {fpn++;}
[-][0-9]*"."[0-9]+ {fnn++;}
. {}
\n {}
%%
void main(int argc,char *argv[])
{
    if(argc!=2){
        printf("Invalid\n");
        exit(0);
    }
    yyin=fopen(argv[1],"r");
    yylex();
    printf("Positive Integers:%d\nNegative Integers:%d\n",pn,nn);
    printf("Positive Fractions:%d\nNegative Fractions:%d\n",fpn,fnn);
}
int yywrap()
{
    return 1;
}
```

Q4) Program to count the no of comment line in a given C program. Also eliminate them and copy that program into separate file.

```
%{
    #include<stdio.h>
    int count=0;
%}
%s COMMENT
%%
"//".* {count++;}
"/*" {BEGIN COMMENT;}
<COMMENT>"*/" {BEGIN 0;count++;}
<COMMENT>\n {count++;}
<COMMENT>. {}
%%
void main(int argc,char*argv[])
{
    if(argc!=3)
    {
        printf("Invalid Arguments\n");
        exit(0);
    }
    yyin=fopen(argv[1],"r");
    yyout=fopen(argv[2],"w");
    yylex();
    printf("No.of comment lines:%d\n",count);
}
int yywrap(){
    return 1;
}
```

5) Program to count the no of ‘scanf’ and ‘printf’ statements in a C program. Replace them with ‘readf’ and ‘writef’ statements respectively

```
%{
    #include<stdio.h>
    int r=0,p=0;
%}
%%
"printf"  {p++;fprintf(yyout,"writef");}
"scanf"  {r++;fprintf(yyout,"readf");}
%%
```

```
void main(int argc,char*argv[])
{
    if(argc!=3)
    {
        printf("Invalid Argument\n");
        exit(0);
    }
    yyin=fopen(argv[1],"r");
    yyout=fopen(argv[2],"w");
    yylex();
    printf("No.of printf statments:%d\n",p);
    printf("No.of scanf statments:%d\n",r);
}
int yywrap(){
    return 1;
}
```

6. Program to recognize a valid arithmetic expression and identify the identifiers and operators present. Print them separately.

```
%{
    #include<stdio.h>
    #include<string.h>
    int valid = 1 ,top=-1,l=0,j=0;
    char OP[10][10],VAL[10][10],a[100],prev=' ',i;
%}
%%
"("|"{"|"[" {
    if(prev!='i'     )
        top++;a[top]=yytext[0];}
    else{
        valid=0;
        return;
    }
")" {
    if(a[top]!='('){valid=0;return;}
    else top--;
}
"}" {
    if(a[top]!='{'){valid=0;return;}
    else top--;
}
"]" {
    if(a[top]!='['){valid=0;return;}
    else top--;
}

"+"|"-"|"*"|"/" {
    if(prev=='+'||prev=='-'||prev=='*'||prev=='/')
    {
        valid=0;
        return;
    }
    prev=yytext[0];
    strcpy(OP[l++],yytext);
}

[0-9]+|[a-zA-Z][a-zA-Z0-9]* {
    prev='i';
    strcpy(VAL[j++],yytext);
}

. {}
\n {}
%%
void main(){
    printf("Enter expression:");
    yylex();
    if(top==-1 && valid==1 && j-l == 1)
    {
        printf("Valid\n");
        printf("Operators:\n");
        for(i=0;i<l;i++)printf("%s\n",OP[i]);
        printf("Operands:\n");
        for(i=0;i<j;i++)printf("%s\n",VAL[i]);
        printf("No. of operators:%d\nNo.of operands:%d\n",l,j);
    }
    else{
        printf("Invalid expression\n");
    }
}
int yywrap(){
    return 1;
}
```

7. Program to recognize whether a given sentence is simple or compound.

```
%{
    #include<stdio.h>
    int isCompound=0;
%}
%%
[ \n\t]+[aA][nN][dD][ \n\t]+ {isCompound=1;}
[ \n\t]+[oO][rR][ \n\t]+ {isCompound=1;}
[ \n\t]+[bB][uU][tT][ \n\t]+ {isCompound=1;}
. {}
\n {}
%%
void main(){
    printf("Enter string:");
    yylex();
    isCompound==1?printf("\nCompound Statement\n") : printf("\nSimple
Statment\n");
}
int yywrap(){
    return 1;
}
```

8. Program to recognize and count the number of identifiers in a given input file.

```
%{
    #include<stdio.h>
    int id=0;
%}
%%
[a-zA-Z][a-zA-Z0-9]* {id++;ECHO;printf("\n");}
. {}
\n {}
%%
void main(int argc,char *argv[])
{
    if(argc!=2)
    {
        printf("Invalid");
        exit(0);
    }
    yyin = fopen(argv[1],"r");
    printf("Identifiers found:\n");
    yylex();
    printf("No. of Identifiers:%d\n",id);
}
int yywrap(){
    return 1;
}
```

YACC

1. Program to test the validity of a simple expression involving operators +, -, * and /

YACC

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%token ID NUM NL
%left '+' '-'
%left '*' '/'
%%
stm : exp NL {printf("Valid Expression");exit(0);}
;
exp : exp '+' exp
| exp '-' exp
| exp '*' exp
| exp '/' exp
| '('exp')'
| ID
| NUM
;
%%
void main()
{
    printf("Enter expression:");
    yyparse();
}
int yyerror()
{
    printf("Invalid");
    exit(0);
}
```

LEX

```
%{
    #include "y.tab.h"
%}
%option noyywrap
%%
\n {return NL;}
[ ]*[0-9]+[ ]* {return NUM;}
[ ]*[a-zA-Z][a-zA-Z0-9]*[ ]* {return ID;}
. {return yytext[0];}
```

2. Program to recognize nested IF control statements and display the levels of nesting.

YACC

```
%{
    #include<stdio.h>
    #include<stdlib.h>
    int count=0;
%}
%token NL S IF ID NUM RELOP
%%
s : if_stmt NL {printf("No.of nested if : %d\n",count);exit(0);}
;
if_stmt : IF  '(' cond ')' '{' if_stmt '}' {count++;}
| S
;
cond : X RELOP X
;
X : ID
| NUM
;
%%
void main(){
    printf("Enter expression:");
    yyparse();
}
int yyerror(){
    printf("Invalid");
    exit(0);
}
```

LEX

```
%{
    #include "y.tab.h"
%}
%option noyywrap
%%
\n {return NL;}
[sS][0-9]* {return S;}
[ \t]*"if"[ \t]* {return IF;}
[0-9]+ {return NUM;}
[a-zA-Z][a-zA-Z0-9]* {return ID;}
">"|"<"|">="|"<="|"=="|"!=" {return RELOP;}
. {return yytext[0];}
```

3. Program to recognize a valid variable, which starts with a letter, followed by any number of letters or digits or underscore.

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%token NL LETTER DIGIT UNDERSCORE
%%
stmt : exp NL {printf("Valid Variable name.\n");exit(0);}
;
exp : LETTER OTHER
;
OTHER : LETTER OTHER
|   DIGIT OTHER
| UNDERSCORE OTHER
|
;
%%
void main(){
    printf("Enter expression:");
    yyparse();
}
int yyerror(){
    printf("Invalid");
    exit(0);
}
```

```
%{
    #include "y.tab.h"
%}
%option noyywrap
%%
\n { return NL;}
[a-zA-Z] {return LETTER;}
[0-9] {return DIGIT;}
"_" {return UNDERSCORE;}
. {return yytext[0];}
```

4. Program to evaluate an arithmetic expression involving operating +, -, * and /.

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%left '+' '-'
%left '*' '/'
%token NL NUM
%%
s : exp NL {printf("Value of expression : %d\n",$1);exit(0);}
;
exp : exp '+' exp   {$$ = $1 + $3;}
| exp '-' exp   {$$ = $1 - $3;}
|exp '*' exp   {$$ = $1 * $3;}
|exp '/' exp   { if($3==0){printf("Divide by zero Error.\n");exit(0);}   $$ =
$1 / $3;}
| '(' exp ')' {$$ = $2;}
| NUM {$$=$1;}
;
%%
void main(){
    printf("Enter expression:\n");
    yyparse();
}
int yyerror(){
    printf("Invalid Expression");
    exit(0);
}
```

```
%{
    #include"y.tab.h"
%}
%option noyywrap
%%
[0-9]+ {yylval= atoi(yytext); return NUM; }
\n {return NL;}
. {return yytext[0];}
```

5. Program to recognize the grammar (a n b, n>=10)

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%token NL A B
%%
stmt : exp NL {printf("Valid Expression\n");exit(0);}
;
exp : A A A A A A A A A A a B
;
a : A a
|
;
%%
void main(){
    printf("Enter expression:");
    yyparse();
}
int yyerror(){
    printf("Invalid expression.");
    exit(0);
}
```

```
%{
    #include "y.tab.h"
%}
%option noyywrap
%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
. {return yytext[0];}
```

6. Program to recognize the grammar (a n b n , n>=0)

```
%{
    #include<stdio.h>
    #include<stdlib.h>
%}
%token NL A B
%%
stmt : exp NL {printf("Valid\n");exit(0);}
;
exp : A exp B
|
;
%%
void main()
{
    printf("Enter expression:");
    yyparse();
}
int yyerror(){
    printf("Invalid");
    exit(0);
}
```

```
%{
    #include "y.tab.h"
%}
%option noyywrap
%%
\n { return NL; }
[aA] {return A;}
[bB] {return B;}
. {return yytext[0];}
```