```
1(a) %{ #include<stdio.h> #include<string.h>
intnooper=0,nooperand=0,top=0,i=0,j=0,tnoope
r=0,tnoopnd=0,valid=0; char
opnd[10][10],opert[10][10]; %}
%% "(" { valid =1;} ")" { valid=0;}
"+"|"*" {nooper++;
strcpy(opert[i],yytext);i++;}
[0-9]+ {nooperand++;
strcpy(opnd[j],yytext);j++;}
[^+*]"("[^0-9] YYFAIL(); %%
int main()
{ int k; printf("Enter the expression\n");
yylex();
if(valid==0 &&(nooperand-nooper)==1)
{ printf("The exp is valid\n");
printf("The operator are\n");
for(k=0;k<i;k++)
printf("%s\n",opert[k]);
printf("The operands are\n");
for(k=0;k<j;k++)
printf("%s\n",opnd[k]); } else
{ printf("The exp is invalid"); return 0; } }
int YYFAIL() { printf("Invalid"); exit(0); }
```

1(b)LEX PART %{ #include "y.tab.h"
#include <stdlib.h> extern int yylval; %}
%% [0-9]+ {yylval=atoi(yytext);
return NUM;} \n {return 0;}
. {return yytext[0];} %%
YACC PART %{ #include<stdio.h> int
valid=1; %} %token NUM
%left '+' '-'
%left '*' '/' %%
Stmt:expr{if(valid)
{ printf("Result=%d\n",$$);} }
expr:expr'+'expr {$$=$1+$3;}
|expr'-'expr {$$=$1-$3;}
|expr'*'expr {$$=$1*$3;}
|expr'/'expr {if($3==0){
valid=0;printf("Divide by zero error\n");}
else $$=$1/$3;} |'('expr')' {$$=$2;}
|NUM {$$=$1;} ; %% main()
{ printf("Enter arithmetic exression:\n");
yyparse(); if(valid==1)
printf("Expression is valid\n");}
int yyerror()
{ printf("Invalid expression\n"); exit(0); }

2.LEX PART %{ #include "y.tab.h" % }
%% a {return A;} b {return B;} \n {return 0;} . {return yytext[0];} %%
YACC PART
%{ #include <stdio.h> int aCount=0,n; %}
%token A %token B
%%
s : X B { if (aCount<n || aCount>n)
{ YYFAIL(); } }
X : X T | T

```
T : A { aCount++;} ;    %%
int main()
{ printf("Enter the value of n \n");
scanf("%d",&n); printf("Enter the string\n");
yyparse();    printf("Valid string\n");  }
int YYFAIL()  {
printf("Invalid count of 'a'\n");
exit(0);  }       int yyerror() {
printf("Invalid string\n");   exit(0);   }
```

5. #include<stdio.h> #include<stdlib.h>
#include<ctype.h>
char op[2],arg1[5],arg2[5],result[5];
void main()
{ FILE *fp1,*fp2;
fp1=fopen("5input.txt","r");
fp2=fopen("5output.txt","w");
while(!feof(fp1)) {
fscanf(fp1,"%s%s%s%s",result,arg1,op,arg2);
if(strcmp(op,"+")==0)
{ fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nADD R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result); }
if(strcmp(op,"*")==0)
{ fprintf(fp2,"\nMOV R0,%s",arg1);

```
fprintf(fp2,"\nMUL R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);  }
if(strcmp(op,"-")==0)
{  fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nSUB R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);  }
if(strcmp(op,"/")==0)
{  fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nDIV R0,%s",arg2);
fprintf(fp2,"\nMOV %s,R0",result);  }
if(strcmp(op,"=")==0)
{ fprintf(fp2,"\nMOV R0,%s",arg1);
fprintf(fp2,"\nMOV %s,R0\n",result);  } }
```

6(a)

```
%{  #include<stdio.h>
int com=0;  %}      %%
"//".* {com++;}
"/*"([^*]|\*+[^*/])*\*+"/" {com++;}
.|\n {fprintf(yyout,"%s",yytext);}    %%
void main(int argc, char *argv[])
{  yyin=fopen(argv[1],"r");
yyout=fopen(argv[2],"w");   yylex();
printf("No of comment lines=%d\n",com);  }
```

6(b) LEX PART

```
%{  #include <stdio.h>
#include "y.tab.h"  %}
%%int|char|bool|float|void|for|do|while|if|else|
return|void|main {printf("keyword is
%s\n",yytext);return KEY;}
[+|-|*|/|=|<|>] {printf("operator is
%s\n",yytext);return OP;}
[a-zA-Z][_a-zA-Z0-9]* {printf("identifier is
%s\n",yytext);return ID;}  . ;   %%
```

YACC PART **%{ #include <stdio.h>**
#include <stdlib.h> int id=0, key=0, op=0; %}
%token ID KEY OP
%% input: ID input { id++; }
| KEY input { key++; }
| OP input {op++;}
| ID { id++; }
| KEY { key++; }
| OP { op++;} ; %%
extern FILE *yyin;
void main(int argc ,char argv)**
{ yyin = fopen(argv[1],"r"); yyparse();
printf("Keywords = %d\nIdentifiers =
%d\noperators = %d\n", key,id, op); }
void yyerror()
{ printf("Not valid"); }

4.Lex Part:

```
%{#include<stdio.h>#include "y.tab.h"%}
%%
"id" {return id;} "+"  {return plus;}
"*"{return star;} "("{return opar;}
")" {return cpar;} . return yytext[0];
\n  return 0;     %%
```

Yacc Part :

```
%{#include<stdio.h>#include<string.h>
extern FILE *yyin;char inp[30],stack[30];
 int inpCount,sCount;   %}
%token id  %token plus  %token star
%token opar  %token cpar
%%
E : E P T {
printf("$%s\t%s$\tREDUCE E -> E+T\n",stack,inp);
sCount -= 3;  stack[sCount++] = 'E';  stack[sCount] = '\0';
}
| T  {
printf("$%s\t%s$\tREDUCE E -> T\n",stack,inp);
sCount -= 1;
stack[sCount++] = 'E'; stack[sCount] = '\0'; }
T : T S F {
printf("$%s\t%s$\tREDUCE T -> T*F \n",stack,inp);
sCount -= 3;
stack[sCount++] = 'T'; stack[sCount] = '\0'; }
| F{
printf("$%s\t%s$\tREDUCE T -> F \n",stack,inp);
sCount -= 1; stack[sCount++] = 'T'; stack[sCount] = '\0'; }
F : O E C {
printf("$%s\t%s$\tREDUCE F -> (E)  \n",stack,inp);
sCount -= 3; stack[sCount++] = 'F';
stack[sCount] = '\0';   }
F : id {   printf("$%s\t%s$\tSHIFT id\n",stack,inp);
```

```
inp[inpCount++] = ' ';  inp[inpCount++] = ' ';
stack[sCount++] = 'i'; stack[sCount++] = 'd';
stack[sCount] = '\0';
printf("$%s\t%s$\tREDUCE F-> id\n",stack,inp);
sCount -= 2; stack[sCount++] = 'F';
stack[sCount] = '\0';  }
O : opar {printf("$%s\t%s$\tSHIFT (\n",stack,inp);
inp[inpCount++] = ' ';  stack[sCount++] = '(';
stack[sCount] = '\0';  }
C : cpar  {printf("$%s\t%s$\tSHIFT )\n",stack,inp);
inp[inpCount++] = ' ';  stack[sCount++] = ')';
stack[sCount] = '\0';  }
P : plus  {printf("$%s\t%s$\tSHIFT +\n",stack,inp);
inp[inpCount++] = ' ';  stack[sCount++] = '+'
stack[sCount] = '\0';  }
S : star   {printf("$%s\t%s$\tSHIFT *\n",stack,inp);
inp[inpCount++] = ' '; stack[sCount++] = '*';
stack[sCount] = '\0'; }
;
%%
void main(){
        printf("Enter the input : \n");
        scanf("%s",inp);
        FILE *fp = fopen("temp.txt","w");
fprintf(fp,"%s",inp); fclose(fp);
yyin = fopen("temp.txt","r");
printf("Stack\tInput\tAction\n");  yyparse();
if(sCount == 1 && stack[sCount-1] == 'E' &&
inp[inpCount]=='\0')
        {printf("$%s\t%s$\tSuccess\n",stack,inp); }  }
int yyerror(){printf("$%s\t%s$\tError\n",stack,inp); }
```