Prime using lex

```
%{
#include<stdlib.h>
int f,k,i;
%}
%%
[0-9]+ {
  k=atoi(yytext);
  if(k>=0){
     f=0;
     for(i=2;i<k;i++)
     if(k%i==0)
       {
         f=1;
         break;
       }
     if(f==0)
       printf("%d is prime\n",k);
     else
       printf("%d is not prime\n",k);
  }
  else
  printf("invalid\n");
 }
%%
int main(){
yylex();
return 0;
}
```

Output

```
5
5 is prime
4
4 is not prime
44
44 is not prime
29
29 is prime
```

Arithmetic expression check using lex and yacc

lex

```
%{
#include"y.tab.h"
%}
%%
[0-9]+ {return number;}
[A-Za-z][A-Za-z0-9]* {return id;}
\n {return 0;}
. {return yytext[0];}
%%
```

yacc

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
%}
%token number id
%left '+' '-'
%left '*' '/'
%%
exp:exp'+'exp|exp'-'exp|exp'*'exp|exp'/'exp|'('exp')'|id|number;
%%
int main(){
printf("Enter exp: \n");
yyparse();
printf("valid expression \n");
}
int yyerror(){
 printf("invalid\n");
 exit(0);
}
```

Output

```
Enter exp:
a+b*c
valid expression

Enter exp:
4444++
invalid
```

Identifier check using lex and yacc

lex

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

[A-Za-z_][_A-Za-z0-9]* {return id;}
\n {return 0;}
. {return yytext[0];}
%%
```

yacc

```
%{
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
%}
%token id
%%
exp: id;
%%
int main(){
printf("Enter: \n");
yyparse();
printf("valid \n");
}
int yyerror(){
 printf("invalid\n");
 exit(0);
}
```

Output

```
Enter:
abc1
valid

Enter:
1ab
invalid
```

Expression evaluation using lex and yacc

lex

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

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

yacc

```
%{
#include<stdio.h>
#include<stdlib.h>
int f=0;
%}
%token num
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%
arexp:exp {printf("result= %d\n",$$);return 0 ;};
exp:exp'+'exp {$$=$1+$3;}|
 exp'-'exp {$$=$1-$3;}|
 exp'*'exp {$$=$1*$3;}|
 exp'/'exp {$$=$1/$3;}|
 '('exp')' {$$=$2;}|
 num {$$=$1;};
%%
int main(){
printf("Enter: \n");
yyparse();
if(f==0)
printf("valid \n");

}
int yyerror(){
 printf("invalid\n");
 f=1;
 exit(0);
}
```

Output

Enter:
1+2*3-4
result= 3
valid

Lexical analyser using c

```
#include<stdio.h>
#include<ctype.h>
#include<string.h>
int main(){
   FILE *input,*output;
   int l=0,j=0,t=0,i,flag;
   char ch,str[20],num[20];
   input=fopen("input.txt","r");
   output=fopen("output.txt","w");
   char
keyword[30][30]={"int","double","float","for","main","if","else","do","while","break","continue"};
   fprintf(output,"Line no. \t Token no. \t \t Token \t\t Lexeme \n\n");

   while(!feof(input)){
      i=0;flag=0;
      ch=fgetc(input);
      if(ch=='+' || ch=='-' ||ch == '*' ||ch =='/'||ch=='='){
      fprintf(output,"%7d\t\t %7d\t\t operator \t %7c\n",l,t,ch);
      t++;
      }
      else if(ch==';' || ch=='{' ||ch == '}' ||ch =='(' ||ch ==')'||ch =='?'||ch =='@'||ch =='!'||ch =='%'
|| ch==',' ||ch=='['||ch==']'){
      fprintf(output,"%7d\t\t %7d\t\t special char\t %7c\n",l,t,ch);
      t++;
      }
        else if(isdigit(ch)){
        num[i++]=ch;
         ch=fgetc(input);
          while(isdigit(ch) && ch!= ' ' && ch!=',' && ch!=';' ){
        num[i]=ch;
      i++;
      ch=fgetc(input);
      }
         num[i]='\0';
          fseek(input, -1, SEEK_CUR);
       fprintf(output,"%7d\t\t %7d\t\t digit\t %7s\n",l,t,num);
       t++;
       }
        else if(isalpha(ch)){
      str[i]=ch;
      i++;
      ch=fgetc(input);
      while(isalnum(ch) && ch!= ' ' && ch!='(' && ch!='[' && ch!='{'){
         str[i]=ch;
      i++;
      ch=fgetc(input);
      }
```

```
        str[i]='\0';
         fseek(input, -1, SEEK_CUR);
        for(j=0;j<=30;j++){
        if(strcmp(str,keyword[j])==0){flag=1;break;}}
        if(flag==1){
          fprintf(output,"%7d\t\t %7d\t\t keyword\t %7s\n",l,t,str);
          t++;
        }
        else{
        fprintf(output,"%7d\t\t %7d\t\t identifier\t %7s\n",l,t,str);
          t++;
        }
         }
         else if(ch=='\n'){l++;}
    }
    fclose(input);fclose(output);return 0;
}
```

Output

Input.txt

```
int main(){
  int a,b=10,c;
  c=a+b;
}
```

output.txt

Line no.	Token no.	Token	Lexeme
0	0	keyword	int
0	1	keyword	main
0	2	special char	(
0	3	special char	)
0	4	special char	{
1	5	keyword	int
1	6	identifier	a
1	7	special char	,
1	8	identifier	b
1	9	operator	=
1	10	digit	10
1	11	special char	,
1	12	identifier	c
1	13	special char	;
2	14	identifier	c
2	15	operator	=
2	16	identifier	a
2	17	operator	+
2	18	identifier	b
2	19	special char	;
3	20	special char	}

First

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char p[10][10],first[10];
int n=0,n1=0;
int inc(char c){
  for(int z=0;z<n;z++)
       if(c==first[z])
         return 1;
  return 0;
}
void fir(char c){



     if(!isupper(c))
         first[n++]=c;
     for(int i=0;i<n1;i++){
          if(p[i][0]==c){

              if(p[i][2]=='$')
                first[n++]='$';
              else if(islower(p[i][2]))
               first[n++]=p[i][2];

         else
          fir(p[i][2]);
           }

     }

}
void main(){

   char c;
   char ch;

   printf("Enter no of productions:\n");
   scanf("%d",&n1);
   printf("Enter productions (epsilon=$)\n");
   for(int i=0;i<n1;i++)
      scanf("%s%c",p[i],&ch);
   int choice=0;
   printf("Enter char :\n");
   scanf("%c",&c);
   do{
```

```
  n=0;
  fir(c);
  printf("first(%c)={",c);
  for(int i=0;i<n;i++)
  {
    if(i!=n-1)
      printf("%c, ",first[i]);
      else
      printf("%c }",first[i]);
  }
  printf("do you want to continue(press 1) and enter next char\n");
     scanf("%d%c",&choice,&c);

  }while(choice==1);

}
```

output

```
Enter no of productions:
5
Enter productions (epsilon=$)
S=A
A=a
S=s
A=$
A=b
Enter char :
S
first(S)={a, $, b, s }do you want to continue(press 1) and enter next char
0
```

Follow

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
char p[10][10],f[10];
int n=0,n1=0;int j,k;
void fir(char c);
void fol(char c);
void fir(char c){
    if(!isupper(c))
        f[n++]=c;
    for(int i=0;i<n1;i++){
         if(p[i][0]==c){
            if(p[i][2]=='$')
               {if(p[j][k+1]=='\0' && p[j][0]!=c)
                 fol(p[j][0]);
               else if(p[j][k+1]==c && p[j][k+2]!='\0')
                fir(p[j][k+2]);
               }
            else if(islower(p[i][2]))
             f[n++]=p[i][2];
       else
        fir(p[i][2]);
         }
    }
}
void fol(char c){
   if(p[0][0]==c)
   f[n++]='$';
   for(j=0;j<n1;j++){
       for( k=2;k<strlen(p[j]);k++){

       if(p[j][k]==c){

           if(p[j][k+1]=='\0')
              fol(p[j][0]);

           else
             fir(p[j][k+1]);
       }
       }
}
void main(){
  char c;
  char ch;

  printf("Enter no of productions:\n");
  scanf("%d",&n1);
```

```
  printf("Enter productions (epsilon=$)\n");
  for(int i=0;i<n1;i++)
    scanf("%s%c",p[i],&ch);
  int choice=0;
  printf("Enter char :\n");
  scanf("%c",&c);
  do{

  n=0;
  fol(c);
  printf("follow(%c)={",c);
  for(int i=0;i<n;i++)
  {
    if(i!=n-1)
      printf("%c, ",f[i]);
      else
      printf("%c }",f[i]);
  }
  printf("do you want to continue(press 1) and enter next char\n");
     scanf("%d%c",&choice,&c);

  }while(choice==1);
}
```

output

```
Enter no of productions:
5
Enter productions (epsilon=$)
S=ABC
B=b
S=A
C=c
B=c
Enter char :
A
follow(A)={b, c, $ }do you want to continue(press 1) and enter next char
0
```

Constant propagation

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
void input();
void output();
void constant();
void change(int p,char *res);
struct expr{
char op[2],op1[5],op2[5],res[5];
int flag;
}arr[10];

int n;
void main(){

   input();constant();output();

}
void input(){

int i;
printf("Enter max no of exp:");
scanf("%d",&n);
printf("Enter input:\n");
for(i=0;i<n;i++){
scanf("%s",arr[i].op);
scanf("%s",arr[i].op1);
scanf("%s",arr[i].op2);
scanf("%s",arr[i].res);

arr[i].flag=0;}
}
void constant(){
  int i,op1,op2,res;
  char op,res1[5];
  for(i=0;i<n;i++){
  if(isdigit(arr[i].op1[0])&&isdigit(arr[i].op2[0])|| strcmp(arr[i].op,"=")==0){

    op1=atoi(arr[i].op1);
      op2=atoi(arr[i].op2);
      op=arr[i].op[0];
      switch(op){
         case '+':res=op1+op2;break;
                case '-':res=op1-op2;break;
                        case '*':res=op1*op2;break;
```

```
                        case '/':res=op1/op2;break;
                                case '=':res=op1;break;

      }
      sprintf(res1,"%d",res);
      arr[i].flag=1;
      change(i,res1);
  }

  }
}
void output(){

int i=0;
printf("\noptimised code is\n");
for(i=0;i<n;i++){

 if(!arr[i].flag){
 printf("\n%s%s%s%s",arr[i].op,arr[i].op1,arr[i].op2,arr[i].res);

 }
}
}
void change(int p,char *res){
int i;
for(i=p+1;i<n;i++){
if(strcmp(arr[p].res,arr[i].op1)==0)
strcpy(arr[i].op1,res);
else if(strcpy(arr[p].res,arr[i].op2)==0)
strcpy(arr[i].op2,res);

}
}
```

output

```
Enter max no of exp:4
Enter input:
= 3 - a
+ a b t1
+ a c t2
+ t1 t2 t3

optimised code is

+3bt1
+3ct2
+t1t2t3
```

Target code generation

```
#include<stdio.h>
#include<string.h>
char op[2],arg1[5],arg2[5],result[5];
void main(){

   FILE *f1,*f2;
   f1=fopen("input1.txt","r");
   f2=fopen("output1.txt","w");
   fscanf(f1,"%s%s%s%s",op,arg1,arg2,result);
   while(!feof(f1)){
   if(strcmp(op,"+")==0)
   {

     fprintf(f2,"\nMOV R0,%s",arg1);
     fprintf(f2,"\nADD R0,%s",arg2);
     fprintf(f2,"\nMOV %s,R0",result);

   }
   if(strcmp(op,"*")==0)
   {

     fprintf(f2,"\nMOV R0,%s",arg1);
     fprintf(f2,"\nMUL R0,%s",arg2);
     fprintf(f2,"\nMOV %s,R0",result);

   }

   if(strcmp(op,"-")==0)
   {

     fprintf(f2,"\nMOV R0,%s",arg1);
     fprintf(f2,"\nSUB R0,%s",arg2);
     fprintf(f2,"\nMOV %s,R0",result);

   }

   if(strcmp(op,"/")==0)
   {

     fprintf(f2,"\nMOV R0,%s",arg1);
     fprintf(f2,"\nDIV R0,%s",arg2);
     fprintf(f2,"\nMOV %s,R0",result);

   }
      if(strcmp(op,"=")==0)
   {
```

```
      fprintf(f2,"\nMOV R0,%s",arg1);

      fprintf(f2,"\nMOV %s,R0",result);
   }
   fscanf(f1,"%s%s%s%s",op,arg1,arg2,result);
     }fclose(f1);fclose(f2);
}
```

Output

Input1.txt

```
* B C A
+ E F G
/ S R T
- Q P R
= X - Z
```

Output1.txt

```
MOV R0,B
MUL R0,C
MOV A,R0
MOV R0,E
ADD R0,F
MOV G,R0
MOV R0,S
DIV R0,R
MOV T,R0
MOV R0,Q
SUB R0,P
MOV R,R0
MOV R0,X
MOV Z,R0
```