Week 2:

Write in week2.l

```
%{
#include<stdio.h>
%}
char [a-zA-Z]
digit [0-9]
id {char}+{{char}|{digit}}*
fun {id}\(\)
number {digit}+{\.{digit}+}*
%%
int|float|char|for|while {printf("%s is a keyword\n", yytext);}
"{"|"}"|"("|")"|"["|"]" {printf("%s is a bracket\n", yytext);}
#.* {printf("%s is a preprocessor directive\n", yytext);}
{char} {printf("%s is a char\n", yytext);}
{digit} {printf("%s is a number\n", yytext);}
id {printf("%s is a id\n", yytext);}
%%
int main(){
FILE *fp;
fp=fopen("ccode.c", "r");
yyin=fp;
yylex();
fclose(fp);
}
int yywrap(){
return 1;
}
```

Output:

```
>> lex.week2.l
>> gcc lex.yy.c -ll -lm -w
>> ./a.out
>> #include<stdio.h>
Int main(){
        printf("Hello World!");
}
```

Week 3:
Write in week3.c

```
#include<stdio.h>
#include<string.h>
char prol[7][10] = {"S","A","A","B","B","C","C"};
char pror[7][10] = {"A","Bb","Cd","aB","@","Cc","@"};
char prod[7][10] = {"S->A","A->Bb","A->Cc","B->aB","B->@","C->Cc","C->@"};
char first[7][10] = {"abcd","ab","cd","a@","@","c@","@"};
char follow[7][10] = {"$","$","$","a$","b$","c$","d$"};
char table[5][6][10];
int numr(char c){
        switch(c){
                case 'S': return 0;
                case 'A': return 1;
                case 'B': return 2;
                case 'C': return 3;
                case 'a': return 0;
                case 'b': return 1;
                case 'c': return 2;
                case 'd': return 3;
                case '$': return 4;
        }
        return(2);
}
int main(){
        int i,j,k;
        for(i=0;i<5;i++){
                for(j=0;j<6;j++){
                        strcpy(table[i][j], "");
                }
        }

        for(i=0;i<7;i++){
                k = strlen(first[i]);
                for(j=0;j<10;j++){
                        if(first[i][j] !='@'){
                                strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1], prod[i]);
                        }
                }
        }

        for(i=0;i<7;i++){
                if(strlen(pror[i])==1){
                    if(pror[i][0]=='@'){
                              k = strlen(follow[i]);
                              for(j=0;j<k;j++){
                                        strcpy(table[numr(pror[i][0])+1][numr(follow[i][j])+1], prod[i]);
                              }
                    }
          }
}

strcpy(table[0][0], " ");
strcpy(table[0][1], "a");
strcpy(table[0][2], "b");
strcpy(table[0][3], "c");
strcpy(table[0][4], "d");
strcpy(table[0][5], "$");
strcpy(table[1][0], "S");
strcpy(table[2][0], "A");
strcpy(table[3][0], "B");
strcpy(table[4][0], "C");
printf("\n......................................................\n");
for(i=0;i<5;i++){
          for(j=0;j<6;j++){
                    printf("%-10s", table[i][j]);
                              if(j==5){
                                        printf("\n......................................................\n");
                              }
          }
}}
```

Output:

```
>> gcc week3.c
./a.out
```

Week 4:
Write in week4.l

```
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+ {yylval.dval=atof(yytext);
return DIGIT;
}
\n|. return yytext[0];
%%
```

Write in week4.y

```
%{
#include<stdio.h>
%}
%union
{
double dval;
}
%token <dval> DIGIT
%type <dval> expr term factor
%%
line:expr'\n'{printf("%g\n",$1);}
expr:expr'+'term{$$=$1+$3;}
|term
;
term:term'*'factor{$$=$1*$3;}
|factor
;
factor:'('expr')'{$$=$2;}
|DIGIT
;
%%
int main(){
yyparse();
}
yyerror(char *s){
printf("%s", s);
}
```

Output:

```
>> lex week4.l
>> yacc -d week4.y
>> gcc lex.yy.c y.tab.c -ll -lm -w
>> ./a.out
3*5
15
```

Week 6:
Write in week6.l

```
%{
#include "y.tab.h"
%}
%%
[a-zA-Z_][a-zA-Z_0-9]* return id;
[0-9]+(\.[0-9]*)? return num;
[+/*] return op;
. return yytext[0];
\n return 0;
%%
int yywrap()
{
return 1;
}
```

Write in week6.y

```
%{
#include<stdio.h>
int valid=1;
%}
%token num id op
%%
start:id'='s';'
s:id x|num x|'-'num x|'('s')'x;
x:op s|'-'s|
;
%%
int yyerror(){
valid=0;
printf("Invalid expression\n");
return 0;
}
int main(){
printf("Enter expression: \n");
yyparse();
if(valid){printf("Valid Expression\n");}}
```

Output:

```
>> lex week6.l
>> yacc -d week6.y
>> gcc lex.yy.c y.tab.c –ll -lm -w
>> ./a.out
a+b=c;
Valid Expression
```