INDEX

S.NO	Program Name	Date	Sign
1.	Introduction to CD **a.** Design a Lexical Analyzer for C. **b.** The Lexical Analyzer should ignore redundant spaces, tabs, and newlines	18-01-2023	
2.	Implement the Lexical Analyzer using Lex.	01-02-2023	
3.	Design a Predictive Parser for a given language.	15-02-2023	
4.	Design LALR bottom-up parser for a given language.	01-03-2023	
5.	Convert BNF rules into YACC form and write code to generate an abstract syntax tree.	25-04-2023	
6.	Construct YACC code to perform Arithmetic Operations.	25-04-2023	
7.	Write a program to generate intermediate code.	02-05-2023	

Programs

Week 1 (18-01-2023):

1) Constant or Not:

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    char  str[50];
    printf("Enter a String: ");
    gets(str);
    if(atoi(str))
       printf("The Given String \"%s\" is CONSTANT\n",str);
    else
       printf("The Given String \"%s\" is NOT CONSTANT\n",str);
}
```

```
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out removing.c
Enter the string
Shravan Teja
Given string is not constant
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out removing.c
Enter the string
123shrvabna
Given string is constant
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out removing.c
Enter the string
ihave done
Given string is not constant
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$
```

2) Checking Comment:

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

void main(){
    char str[50];
    printf("Enter Input: ");
    gets(str);
```

```
    if(str[0]=='/'){
     if(str[1]=='/')
          printf("Given Statement is a COMMENT\n");
     else if(str[1]=='*'){
          int flag=0,n=strlen(str)-1;
          if(str[n]=='/' && str[n-1]=='*')
               printf("\nGiven Statement is a COMMENT\n");
          else{
               printf("\nGiven Statement is NOT a Comment\n");
          }
     }
     else
          printf("\nGiven Statement is NOT a COMMENT\n");
    }
    else
     printf("\nGiven Statement is NOT a COMMENT\n");
}
```

```
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
//Shravan Teja
The given statement is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
/* Its raining today */
The given statemennt is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
This is a test program
The given statement is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$
```

3) Checking Identifier:

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

void main(){
    char str[50];
    printf("Enter Input: ");
```

```
    gets(str);
    int flag=0;
    if(isalpha(str[0])||str[0]=='_'){
       for(int i=0;i<strlen(str);i++){
            if(isdigit(str[i])|| isalpha(str[i])|| str[i]=='_')
                 flag=1;
            else
                 break;
       }
    }
    if(flag==1)
       printf("This is a Valid Identifier\n");
    else
       printf("This is an Invalid Identified\n");
}
```

```
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
//Shravan Teja
The given statement is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
/* Its raining today */
The given statemennt is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out comment_or_not.c
Enter the input :
This is a test program
The given statement is a comment
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$
```

4) Checking Keyword:

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

void main(){
```

```
    char
keyw[21][10]={"auto","double","struct","break","else","long","swi
tch","case","enum","register","typedef","char","extern","return",
"union","const","float","short","do","if","while"};
    char str[100],tokens[25][25];
    int j=0,k=0,wc=0,flag=0;
    printf("Enter the C Statement: ");
    gets(str);
    for(int i=0;i<strlen(str);i++){
        if(str[i]!=' '){
              tokens[j][k]=str[i];
              k++;
        }
        if(str[i]==' '){
              tokens[j][k]='\0';
              j++;
              k=0;
              wc++;
        }
    }
    tokens[j][k]='\0';
    for(int i=0;i<wc;i++){
        for(int j=0;j<32;j++){
              if(strcmp(keyw[j],tokens[i])==0){
                    printf("\"%s\" is a KeyWord\n",tokens[i]);
                    flag=1;
              }
        }
    }
    if(flag==0)
        printf("There are NO KeyWords\n");
}
```

```
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out keywords_or_not.c
enter the c statement:
long
long is a keyword
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out keywords_or_not.c
enter the c statement:
auto
auto is a keyword
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$
```

5) Checking Operator:

```
#include<stdio.h>
#include<stdlib.h>
void main(){
    char str[5];
    printf("Enter any Operator (+,-,..): ");
    gets(str);
    switch(str[0]){
        case'>':
            if(str[1]=='=')
                printf("Greater than or Equal to\n");
            else
                printf("Strictly Greater than\n");
            break;
        case'<':
            if(str[1]=='=')
                printf("Less than or Equal to\n");
            else
                printf("Strictly Less than\n");
            break;
        case'=':
            if(str[1]=='=')
                printf("Equal to\n");
            else
                printf("Assignment\n");
            break;
        case'!':
            if(str[1]=='=')
```

```
                printf("Not Equal\n");
            else
                printf("Bit NOT\n");
            break;
        case'&':
            if(str[1]=='&')
                printf("Logical AND\n");
            else
                printf("Bitwise AND\n");
            break;
        case'|':
            if(str[1]=='|')
                printf("Logical OR\n");
            else
                printf("Bitwise OR\n");
            break;
        case'+':
            printf("Addition\n");
            break;
        case'-':
            printf("Substraction\n");
            break;
        case'*':
            printf("Multiplication\n");
            break;
        case'/':
            printf("Division\n");
            break;
        case'%':
            printf("Modulus\n");
            break;
        default:
            printf("Entered value \"%s\" is NOT an
Operator\n",str);
    }
}
```

```
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out Checking_operators.c
Enter the string
/

 Divisionvnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903$ ./a.out Checking_operators.c
Enter the string
+
```

Week 2 (01-02-2023):

LEX file week 2.l:

```
%{
#include<stdio.h>
%}
digit [0-9]+
word [A-Za-z]+
spsym [(){};,%\{\]]
arith [+-/*]
whitspc[ \t\n]
underscr[_]
%%
{whitspc}+ ;
\"[^\n\"]*\" {printf("\n %s is a literal",yytext);}
int |
include |
if  |
else |
while |
do |
switch |
case |
default |
break |
continue |
scanf {printf("\n%s is a Keyword",yytext);}
{spsym} {printf("\n%s is a Special Symbol",yytext);}
 {arith} {printf("\n%s is a Binary Operator",yytext);}
= {printf("\n%s is a Assignment operator",yytext);}
"++" |"--" {printf("\n%s is an Unary Operator",yytext);}
"&" |"|" |"^" {printf("\n %s is bitwise operator",yytext);}
"<" |">" |"<=" |">=" |"==" |"!=" {printf("\n %s is a relational
operator",yytext);}
{digit}+ {printf("\n %s is an integer constant",yytext);}
({digit}+\.{digit}*)|({digit}*\.{digit}+) {printf("\n %s is a
float constant",yytext);}
({word}({word}|{digit}|{underscr})*) {printf("\n%s is a
Identifier",yytext);}
%%
int main(int argc,char *argv[])
{
FILE *fp;
fp=fopen(argv[1],"r");
```

```
    if(!fp)
    {
    printf("cnt open:%s",argv[1]);
    exit(1);
    }
    yyin=fp;
    yylex();
    }
    int yywrap()
    {
    return 1;
    }
```

C file week 2.c:

```
#include<stdio.h>
#include<conio.h>
void main(){
     int a,b,c;
     a=1;
     b=2;
     c=a+b;
     printf("Sum:%d",c);
}
```

```
 is  a white Spacevnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903/Week-2$ lex week_21.l
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903/Week-2$ cc lex.yy.c
vnrvjiet@vnrvjiet-HP-ProDesk-400-G7-Microtower-PC:~/Desktop/21075A6903/Week-2$ ./a.out var.c

# is a Special Symbol
include is a Keyword
 < is a relational operator
stdio is a Identifier
. is a Special Symbol
h is a Identifier
 > is a relational operator

 is  a white Space
void is a Identifier
   is  a white Space
main is a Identifier
( is a Special Symbol
) is a Special Symbol
{ is a Special Symbol

        is  a white Space
int is a Keyword
   is  a white Space
a is a Identifier
, is a Special Symbol
b is a Identifier
, is a Special Symbol
c is a Identifier
; is a Special Symbol

        is  a white Space
a is a Identifier
= is a Assignment operator
 1 is an integer constant
; is a Special Symbol
```

Week 3 (15-02-2023):

Predictive Parsing:

```
#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->Cd", "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], " ");
```

```
  printf("The following grammar is used for Parsing Table:\n");

  for (i = 0; i < 7; i++)
     printf("%s\n", prod[i]);
  printf("\nPredictive parsing table:\n");
  fflush(stdin);
  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(prol[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----------------------------------------------------------------------------------------------------
);
} }
```

```
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ gcc week3.c
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ ./a.out
The following grammar is used for Parsing Table:
S->A
A->Bb
A->Cd
B->aB
B->@
C->Cc
C->@

Predictive parsing table:

--------------------------------------------------------
          a         b         c         d         $
--------------------------------------------------------
S         S->A      S->A      S->A
--------------------------------------------------------
A         A->Bb     A->Cd     A->Cd
--------------------------------------------------------
B         B->aB     B->@                          B->@
--------------------------------------------------------
C                   C->@      C->Cc     C->@      C->@
--------------------------------------------------------
```

Week 4 (25/04/2023):

LALR Parsing:

LEX file Week4.l :

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

 %%
```

YACC file Week 4.y:

```
%{
/*This YACC specification file generates the LALR parser for the
program
considered in experiment 4.*/
#include<stdio.h>
%}
%union
{
double dval;
}
%token <dval> DIGIT
%type <dval> expr
%type <dval> term
%type <dval> 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);
}
```

```
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ lex week4.l
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ yacc -d week4_1.y
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ cc lex.yy.c y.tab.c -ll -lm
y.tab.c: In function ‘yyparse’:
y.tab.c:1225:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-
function-declaration]
 1225 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1378:7: warning: implicit declaration of function ‘yyerror’; did you me
an ‘yyerrok’? [-Wimplicit-function-declaration]
 1378 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
week4_1.y: At top level:
week4_1.y:27:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   27 | yyerror(char *s)
      | ^~~~~~~
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ ./a.out
2+3
5
4*5
syntax errorubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ ./a.out
4*5
20
^C
```

Week 5 (25/04/2023):

Week5.l:

```
%{
#include"y.tab.h"
#include<stdio.h>
#include<string.h>
int LineNo=1;
%}
identifier [a-zA-Z][_a-zA-Z0-9]*
number [0-9]+|([0-9]*\.[0-9]+)
%%
main\(\) return MAIN;
if return IF;
else return ELSE;
while return WHILE;
int |
char |
float return TYPE;
{identifier} {strcpy(yylval.var,yytext);
return VAR;}
{number} {strcpy(yylval.var,yytext);
return NUM;}
\< |
\> |
\>= |
\<= |
== {strcpy(yylval.var,yytext);
return RELOP;}
[ \t] ;
\n LineNo++;
. return yytext[0];
%%
```

Week5.y:

```
%{
#include<string.h>
#include<stdio.h>
struct quad
{
char  op[5];
char arg1[10];
```

```
char arg2[10];
char result[10];
}QUAD[30];

struct stack
{
int items[100];
int top;
}stk;
int Index=0,tIndex=0,StNo,Ind,tInd;
extern int LineNo;
%}
%union
{
char var[10];
}
%token <var> NUM VAR RELOP
%token MAIN IF ELSE WHILE TYPE
%type <var> EXPR ASSIGNMENT CONDITION IFST ELSEST WHILELOOP
%left '-' '+'
%left '*' '/'
%%
PROGRAM : MAIN BLOCK
;
BLOCK: '{' CODE '}'
;
CODE: BLOCK
| STATEMENT CODE
| STATEMENT
;

STATEMENT: DESCT ';'
| ASSIGNMENT ';'
| CONDST
| WHILEST
;
DESCT: TYPE VARLIST
;
VARLIST: VAR ',' VARLIST
| VAR
;
```

```
ASSIGNMENT: VAR '=' EXPR{
strcpy(QUAD[Index].op,"=");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,$1);
strcpy($$,QUAD[Index++].result);
}
;
EXPR: EXPR '+' EXPR {AddQuadruple("+",$1,$3,$$);}
| EXPR '-' EXPR {AddQuadruple("-",$1,$3,$$);}
| EXPR '*' EXPR {AddQuadruple("*",$1,$3,$$);}
| EXPR '/' EXPR {AddQuadruple("/",$1,$3,$$);}
| '-' EXPR {AddQuadruple("UMIN",$2,"",$$);}
| '(' EXPR ')' {strcpy($$,$2);}
| VAR
| NUM
;

CONDST: IFST{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
}
| IFST ELSEST
;
IFST: IF '(' CONDITION ')' {
strcpy(QUAD[Index].op,"==");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"FALSE");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
BLOCK { strcpy(QUAD[Index].op,"GOTO"); strcpy(QUAD[Index].arg1,"");
strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
};
ELSEST: ELSE{
```

```
tInd=pop();
Ind=pop();

push(tInd);
sprintf(QUAD[Ind].result,"%d",Index);
}
BLOCK{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
};
CONDITION: VAR RELOP VAR {AddQuadruple($2,$1,$3,$$);
StNo=Index-1;
}
| VAR
| NUM
;
WHILEST: WHILELOOP{
Ind=pop();
sprintf(QUAD[Ind].result,"%d",StNo);
Ind=pop();
sprintf(QUAD[Ind].result,"%d",Index);
}
;
WHILELOOP: WHILE'('CONDITION ')' {
strcpy(QUAD[Index].op,"==");
strcpy(QUAD[Index].arg1,$3);
strcpy(QUAD[Index].arg2,"FALSE");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
BLOCK {
strcpy(QUAD[Index].op,"GOTO");
strcpy(QUAD[Index].arg1,"");

strcpy(QUAD[Index].arg2,"");
strcpy(QUAD[Index].result,"-1");
push(Index);
Index++;
}
;
```

```
%%
extern FILE *yyin;
int main(int argc,char *argv[])
{
FILE *fp;
int i;
if(argc>1)
{
fp=fopen(argv[1],"r");
if(!fp)
{
printf("\n File not found");
exit(0);
}
yyin=fp;
}
yyparse();
printf("\n\n\t\t ----------------------------""\n\t\t Pos Operator
\tArg1 \tArg2 \tResult" "\n\t\t ------------------- ");
for(i=0;i<Index;i++)
{
printf("\n\t\t %d\t %s\t %s\t
%s\t%s",i,QUAD[i].op,QUAD[i].arg1,QUAD[i].arg2,QUAD[i].result);
}
printf("\n\t\t--------------------------------");
printf("\n\n"); return 0; }
void push(int data)
{ stk.top++;

if(stk.top==100)
{
printf("\n Stack overflow\n");
exit(0);
}
stk.items[stk.top]=data;
}
int pop()
{
int data;
if(stk.top==-1)
{
```

```
printf("\n Stack underflow\n");
exit(0);
}
data=stk.items[stk.top--];
return data;
}
void AddQuadruple(char op[5],char arg1[10],char arg2[10],char
result[10])
{
strcpy(QUAD[Index].op,op);
strcpy(QUAD[Index].arg1,arg1);
strcpy(QUAD[Index].arg2,arg2);
sprintf(QUAD[Index].result,"t%d",tIndex++);
strcpy(result,QUAD[Index++].result);

}
yyerror()
{
printf("\n Error on line no:%d",LineNo);
}
```

Week5.c:

```
main() {
  int a, b, c;
  if (a < b) {
    a = a + b;
  }
  while (a < b) {
    a = a + b;
  }
  if (a <= b) {
    c = a - b;
  } else {
    c = a + b;
  }
}
```

```
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ lex week5.l
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ yacc -d week5.y
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ gcc lex.yy.c y.tab.c -ll -lm -w
ubuntu-nslab@ubuntunslab:~/Desktop/21075A6903$ ./a.out week5.c


                ----------------------------
                Pos Operator   Arg1    Arg2    Result
               -------------- -----
                0       <       a       b       t0
                1       ==      t0      FALSE   5
                2       +       a       b       t1
                3       =       t1              a
                4       GOTO                    5
                5       <       a       b       t2
                6       ==      t2      FALSE   10
                7       +       a       b       t3
                8       =       t3              a
                9       GOTO                    5
                10      <=      a       b       t4
                11      ==      t4      FALSE   15
                12      -       a       b       t5
                13      =       t5              c
                14      GOTO                    17
                15      +       a       b       t6
                16      =       t6              c
                ----------------------
```

Week 6 (25/04/2023):

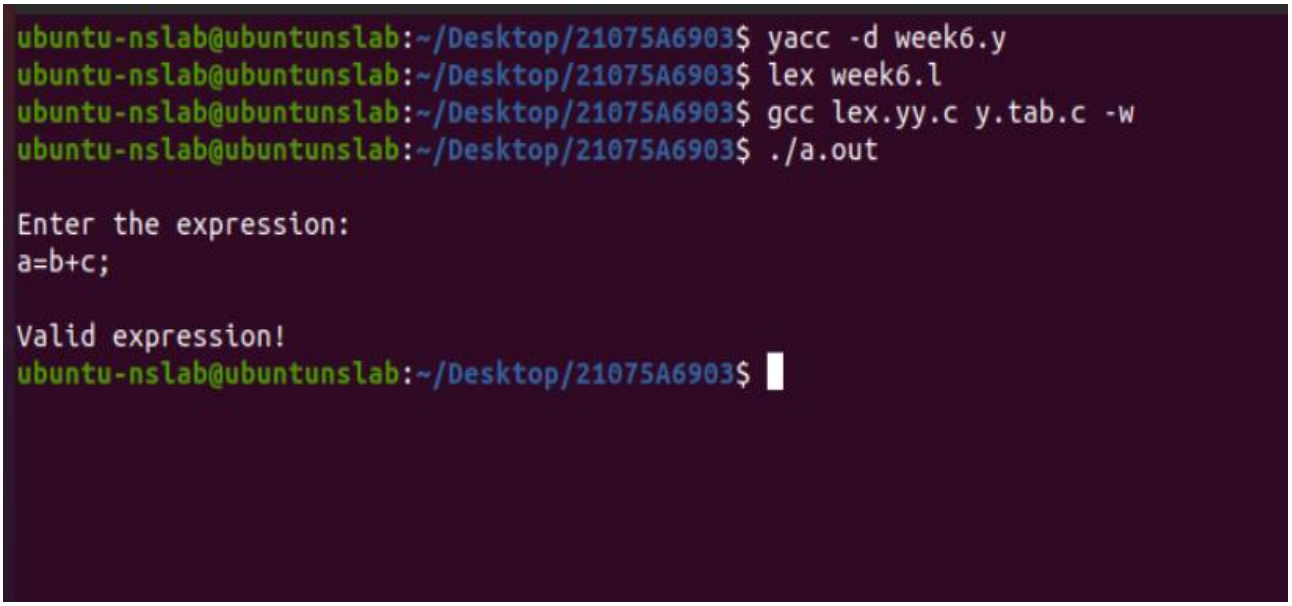
Week6.l:

```
%{
#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;
}
```

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("\nInvalid expression!\n");
return 0;
```

```
}
int main()
{
printf("\nEnter the expression:\n");
yyparse();
if(valid)
{
printf("\nValid expression!\n");
}
}
```



Week 7 (25/04/2023):

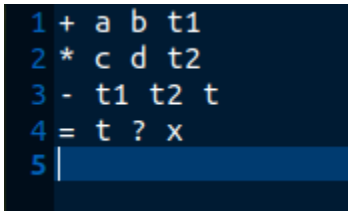
Week7.c:

```
#include <stdio.h>
#include <curses.h>
#include <string.h>
char op[2], arg1[5], arg2[5], result[5];
void main()  {
  FILE *fp1, *fp2;
  fp1 = fopen("week7inp.txt", "r");
  fp2 = fopen("week7out.txt", "w");
  while (!feof(fp1)) {
   fscanf(fp1, "%s%s%s%s", op, arg1, arg2, result);
   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", result);
   }
  }
  fclose(fp1);
```

```
  fclose(fp2);
  getch();
}
```

Week7inp.txt:

```
+ a b t1
* c d t2
- t1 t2 t
= t ? x
```



Week7out.txt:

```
MOV R0,a
ADD R0,b
MOV t1,R0
MOV R0,c
MUL R0,d
MOV t2,R0
MOV R0,t1
SUB R0,t2

MOV t,R0
MOV R0,t
MOV x,R0
MOV R0,t
MOV x,R0
```