Ex-15:

calc.l:

```
%{
#include<stdio.h>
#include "calc.tab.h"
extern int yylval;
%}
%%
[0-9]+ {
yylval=atoi(yytext);
return NUMBER;
}
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

calc.y:

```
%{
#include<stdio.h>
#include<stdlib.h>
int flag=0;
void yyerror(const char *s);
int yylex(void);
%}

%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'

%%

ArithmeticExpression: E{
printf("\nResult=%d\n",$$);
return 0;
}
;
```

```
E:E'+'E {$$=$1+$3;}
|E'-'E {$$=$1-$3;}
|E'*'E {$$=$1*$3;}
|E'/'E {$$=$1/$3;}
|E'%'E {$$=$1%$3;}
|'('E')' {$$=$2;}
| NUMBER {$$=$1;}
;

%%
int main()
{
printf("\nEnter Any Arithmetic Expression which can have operations Addition,
Subtraction,Multiplication, Divison, Modulus and Round brackets:\n");
yyparse();
if(flag==0)
printf("\nEntered arithmetic expression is Valid\n\n");
}

void yyerror(const  char *s)
{
printf("\nEntered arithmetic expression is Invalid\n\n");
flag=1;
}
```

```
        l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ bison -d calc.y
calc.y:31 parser name defined to default :"parse"
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ flex calc.l
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ gcc -o calc calc.tab.c lex.yy.c -lfl
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ ./calc

Enter Any Arithmetic Expression which can have operations Addition, Subtraction,Multiplication,
Divison, Modulus and Round brackets:
3+5*(2*9)

Result=93

Entered arithmetic expression is Valid
```

ex-14:

```
#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 1;
      case 'b': return 2;
      case 'c': return 3;
      case 'd': return 4;
      case '$': return 5;
   }
   return -1;  // Return -1 for invalid characters
}

int main() {
   int i, j, k;

   // Initialize table with empty strings
   for (i = 0; i < 5; i++)
      for (j = 0; j < 6; j++)
         strcpy(table[i][j], " ");

   printf("\nThe following is the predictive parsing table for the following grammar:\n");
```

```
for (i = 0; i < 7; i++)
   printf("%s\n", prod[i]);

printf("\nPredictive parsing table is\n");
for (i = 0; i < 7; i++) {
   k = strlen(first[i]);
   for (j = 0; j < k; j++) {
      if (first[i][j] != '@') {
         int row = numr(prol[i][0]);
         int col = numr(first[i][j]);
         if (row >= 0 && col >= 0)
            strcpy(table[row + 1][col + 1], prod[i]);
      }
   }
}

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

// Set up table headers
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");
        }
    }
    return 0;
}
```

ex-12:

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

int E(), Edash(), T(), Tdash(), F();
char *ip;
char string[50];

int main() {
    printf("Enter the string\n");
    scanf("%s", string);
```

```
    ip = string;

    printf("\n\nInput\tAction\n----------------------------------\n");
    if (E() && *ip == '\0') {
        printf("\n----------------------------------\n");
        printf("\n String is successfully parsed\n");
    } else {
        printf("\n----------------------------------\n");
        printf("Error in parsing String\n");
    }
}

int E() {
    printf("%s\tE->TE' \n", ip);
    if (T()) {
        if (Edash()) {
            return 1;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
}

int Edash() {
    if (*ip == '+') {
        printf("%s\tE'->+TE' \n", ip);
        ip++;
        if (T()) {
            if (Edash()) {
```

```
            return 1;
        } else {
            return 0;
        }
    } else {
        return 0;
    }
  } else {
    printf("%s\tE'->^ \n", ip);
    return 1;
  }
}

int T() {
  printf("%s\tT->FT' \n", ip);
  if (F()) {
    if (Tdash()) {
      return 1;
    } else {
      return 0;
    }
  } else {
    return 0;
  }
}

int Tdash() {
  if (*ip == '*') {
    printf("%s\tT'->*FT' \n", ip);
    ip++;
    if (F()) {
```

```
            if (Tdash()) {
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    } else {
        printf("%s\tT'->^ \n", ip);
        return 1;
    }
}

int F() {
    if (*ip == '(') {
        printf("%s\tF->(E) \n", ip);
        ip++;
        if (E()) {
            if (*ip == ')') {
                ip++;
                return 1;
            } else {
                return 0;
            }
        } else {
            return 0;
        }
    } else if (*ip == 'i') {
        ip++;
        printf("%s\tF->id \n", ip);
```

```
        return 1;
    } else {
        return 0;
    }
}
```

Ex-9:

```
%{
#include <stdio.h>
#include <string.h>
int i = 0;
%}

%%

[a-zA-Z0-9]+ { i++; }
\n { printf("%d\n", i); i = 0; }
. { /* Ignore other characters */ }

%%

int yywrap(void) {
    return 1;
}

int main() {
    yylex();
    return 0;
}
```

output:

```
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ flex lex.l
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ gcc lex.yy.c -o lex -lfl
l4-11@l411-OptiPlex-3090:~/Desktop/OS82$ ./lex
Navya Sree
2
```

ex-7:

```
#include<stdio.h>
#include<string.h>
int main(){
char input[30];
int j=0;
printf("enter string(0+1)*00:");
scanf("%s",input);
int c=0,state=0;
if(strlen(input)<2){
printf("string not accepted");
} else{
for(j=0;j<strlen(input);j++){
if(input[j]=='0'||input[j]=='
            c = c + 1;
          }
       }

       if (c == strlen(input)) {

          if (input[strlen(input) - 1] == '0') {
             state = 1;
          }
          if (input[strlen(input) - 2] == '0' && state == 1) {
             state = 2;
          } else {
             state = -1;
          }
          if (state == 2) {
             printf("String accepted");
           } else {
```

```
            printf("String not accepted");
        }
    } else {
        printf("String not accepted");
    }
  }
  return 0;
}
```

output:

```
l4-11@l411-OptiPlex-3090:~/
enter string(0+1)*00:10100
String accepted
```

Ex-6:

```
#include<stdio.h>
#include<string.h>
int main(){
char array[20];
  printf("Enter the input string aa*bb*:");
  scanf("%s", array);

  int state = 0;
  int i = 0;

  if (array[i] == 'a') {
    state = 1;
  } else {
    state = -1;
  }
```

```
if (state != -1) {
   for (i = 1; i < strlen(array); i++) {

      if (array[i] != 'a' && array[i] != 'b') {
         state = -1;
         break;
      }

      if (state == 1) {
         if (array[i] == 'a') {
            state = 1;
         } else {
            if (array[i] == 'b') {
               state = 2;
            } else {
               state = -1;
               break;
            }
         }
      } else if (state == 2) {
         if (array[i] == 'b') {
            state = 2;
         } else {
            state = -1;
            break;
         }
      }
   }
}
```

```
    if (state == 2 && i == strlen(array)) {
        printf("string accepted");
    } else {
        printf("string not accepted");
    }
    return 0;
}
```