# System Software

Nehal Jhajharia (U20CS093)
Lab Assignment 8

1. Write a YACC and LEX program to implement a Calculator and recognize a valid arithmetic expression that uses operator +, -, *, /.

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

/* Rule Section */
%%
[0-9]+ {
          yylval=atoi(yytext);
          return NUMBER;

}
[\t] ;

[\n] return 0;

. return yytext[0];

%%

int yywrap()
{
return 1;
}

// commands
/*
flex 1.l
yacc 1.y -d
gcc lex.yy.c y.tab.c -w -ll
./a.out
*/

%{
#include<stdio.h>
int flag=0;
%}

%token NUMBER

%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;}

| NUMBER {$$=$1;}

;

%%

void main()
{
   printf("\nEnter Arithmetic Expression\n");

   yyparse();
   if(flag==0) {
      printf("\nEntered arithmetic expression is Valid\n\n");
   }
}

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

2. Write a YACC and LEX program to check whether a given string is palindrome or not.

```
%{
/* Definition section */
#include <stdio.h>
#include <stdlib.h>
#include "y.tab.h"
%}

/* %option noyywrap */
/* Rule Section */
%%

[a-zA-Z]+ {yylval.f = yytext; return STR;}
[-+()*/] {return yytext[0];}
[ \t\n]   {;}

%%

int yywrap()
{
return -1;
}

%{
/* Definition section */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
extern int yylex();

void yyerror(char *msg);
int flag;

int i;
int k =0;
%}

%union {
char* f;
}

%token <f> STR
%type <f> E

/* Rule Section */
```

```
%%

S : E {
          flag = 0;
          k = strlen($1) - 1;
          if(k%2==0){
          for (i = 0; i <= k/2; i++) {
             if ($1[i] != $1[k-i]) {
                flag = 1;
             }
          }
          if (flag == 1) printf("Not palindrome\n");
          else printf("palindrome\n");
          printf("%s\n", $1);
          }else{
          for (i = 0; i < k/2; i++) {
          if ($1[i] == $1[k-i]) {
          } else {
             flag = 1;
             }
          }
          if (flag == 1) printf("Not palindrome\n");
          else printf("palindrome\n");
          printf("%s\n", $1);
          }
}
;

E : STR {$$ = $1;}
;

%%

void yyerror(char *msg)
{
fprintf(stderr, "%s\n", msg);
exit(1);
}

//driver code
int main()
{
yyparse();
return 0;
}
```

3. Write a program for implementing given grammar for computing the expression using semantic rules of the YACC tool and LEX.

Grammar: S-> SS* | SS+ | a

```
%{
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}
%%
[*] {yylval = 1;return A;}
[+] {yylval = 2;return B;}

a {yylval = 3;return C;}
\n {yylval = 4;return 0;}
%%
int yywrap()
{
return 1;
}

%{
#include<stdio.h>
#include<stdlib.h>
int yyerror(char *msg)
{
printf("Invalid string\n");
exit(0);
}
%}
%token A B C
%%
R : S {printf("Valid string\n");}
;
S : S S B
| S S A
| C
;
%%
int main()
{
printf("Enter the string: ");
yyparse();
return 0;
}
```

4. Write a YACC and LEX program to accept strings that start and ends with 0 or 1.

```
%{
/* Definition section */
extern int yylval;
#include "y.tab.h"
%}

/* Rule Section */
%%

0 {yylval = 0; return ZERO;}
```

```
1 {yylval = 1; return ONE;}

.|\n {yylval = 2; return 0;}

%%

/*
flex 4.l
yacc 4.y -d
gcc y.tab.c -lfl -ly
./a.out
*/

%{
/* Definition section */
#include<stdio.h>
#include <stdlib.h>
void yyerror(const char *str)
{
printf("\nSequence Rejected\n");
}

%}

%token ZERO ONE

/* Rule Section */
%%

r : s {printf("\nSequence Accepted\n\n");}
;

s : n
| ZERO a
| ONE b
```

```
;

a : n a
| ZERO
;

b : n b
| ONE
;

n : ZERO
| ONE
;

%%

#include"lex.yy.c"
//driver code
int main()
{
printf("\nEnter Sequence of Zeros and Ones : ");
yyparse();
printf("\n");
return 0;
}
```