4.a. YACC PROGRAM TO RECOGNIZE A VALID ARITHMETIC EXPRESSION THAT USE OPERATOR +,-,*,/

PROGRAM:

```
%{ /* validate simple arithmetic expression */
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#include<string.h>
#define YYSTYPE double
%}
%token num
%left '+' '-'
%left '*' '/'
%%
st: st expr '\n' {printf("Valid");}
|st '\n'
|
|error '\n' {printf("INVALID");}
;
expr: num
|expr '+' expr
|expr '/' expr
%%
main()
{
printf(" ENTER AN EXPRESSION TO VALIDATE");
yyparse();
}
yylex()
{
int ch;
while((ch=getchar())==' ');
if(isdigit(ch)|ch=='.')
{
ungetc(ch,stdin);
scanf("%lf",&yylval);
return num;
}
return ch;
}
yyerror(char *s)
{
printf("%S",s);
}
```

4.b. YAAC PROGRAM TO RECOGNIZE THAT STARTS WITH A LETTER FOLLOWED BY NUMBER OR DIGITS

PROGRAM:

```
%{ /* Y prg to recognize valid variable, which starts with a letter,
followed by any number of letters or digits. */
#include<stdio.h>
#include<ctype.h>
%}
%token let dig
%%
sad: let recld '\n' {printf("accepted\n"); exit(0);}
| let '\n' {printf("accepted\n"); exit(0);}
|
|error {yyerror("rejected\n");}
;
recld: let recld
| dig recld
| let
| dig
;
%%
yylex()
{
char ch;
while((ch=getchar())==' ');
if(isalpha(ch))
return let;
if(isdigit(ch))
return dig;
return ch;
}
yyerror(char *s)
{
printf("%s",s);

}
main()
{
printf("ENTER A variable : ");
yyparse();
}
```

4.c.IMPLEMENTATION OF CALCULATOR USING LEX AND YAAC

cal.l

```
%{
#include <stdlib.h>
#include <stdio.h>
#include "y.tab.h"
void yyerror(char*);
extern int yylval;
%}
%%
[ \t]+ ;
[0-9]+ {yylval = atoi(yytext);
return INTEGER;}
[-+*/] {return *yytext;}
"(" {return *yytext;}
")" {return *yytext;}
\n {return *yytext;}
. {char msg[25];
sprintf(msg,"%s <%s>","invalid character",yytext);
yyerror(msg);
}
```

cal.y

```
%{
#include <stdlib.h>
#include <stdio.h>
int yylex(void);
#include "y.tab.h"
%}
%token INTEGER
%%
program:
line program
| line
line:
expr '\n' { printf("%d\n",$1); }
| 'n'
expr:
expr '+' mulex { $$ = $1 + $3; }
| expr '-' mulex { $$ = $1 - $3; }
| mulex { $$ = $1; }
mulex:
mulex '*' term { $$ = $1 * $3; }
| mulex '/' term { $$ = $1 / $3; }
| term { $$ = $1; }
```

```
term:
'(' expr ')' { $$ = $2; }
| INTEGER { $$ = $1; }
%%
void yyerror(char *s)
{
fprintf(stderr,"%s\n",s);
return;
}
yywrap()
{
return(1);
}
int main(void)
{
yyparse();
return 0;
}
```