Exercise 8

Infix to Postfix and Prefix

Infix to Postfix (Identfier Expression)

Code

.l

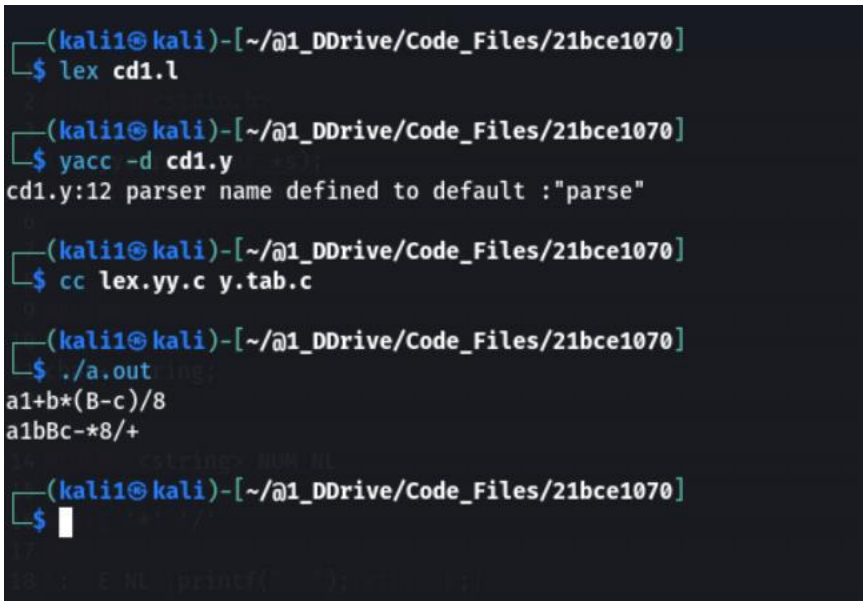
```
%{
#include"y.tab.h"
extern YYSTYPE yylval;
%}
%%
[A-Za-z0-9]+  {yylval.string=strdup(yytext); return NUM;}
\n      return NL;
.       return *yytext;
%%
int yywrap(){
   return 1;
}
```

.y

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

%}
```

```
%union
{
char* string;
}

%token <string> NUM NL
%left '+' '-'
%left '*' '/'
%%
S:  E NL {printf("\n");return 0;}
   ;
E:  E '+' E {printf("+");}
   |  E '*' E {printf("*");}
   |  E '-' E {printf("-");}
   |  E '/' E {printf("/");}
   |  '(' E ')'
   |  NUM     {printf("%s", $1);}
   ;
%%

int main(){
   yyparse();
}

int yyerror (char *msg) {
   return printf ("error YACC: %s\n", msg);
}
```

Output



Infix to prefix (identifier expression)

```
.l
%{

   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>
   #include "y.tab.h"
%}

INTEGER    [0-9]+
IDENTIFIER [_a-zA-Z][_a-zA-Z0-9]*
%%

exit.*      { return EXIT; }
```

```
quit.*      { return EXIT; }
{INTEGER}   { yylval.exp = strdup(yytext);  return INTEGER;}
{IDENTIFIER}   { yylval.exp = strdup(yytext);  return IDENTIFIER;}
[+-]        { yylval.exp = strdup(yytext);  return OPR1;  }
[/*]        { yylval.exp = strdup(yytext);  return OPR2;  }
[()]        { return yytext[0]; }
\n          { return NEWLINE; }
.           ;

%%
```

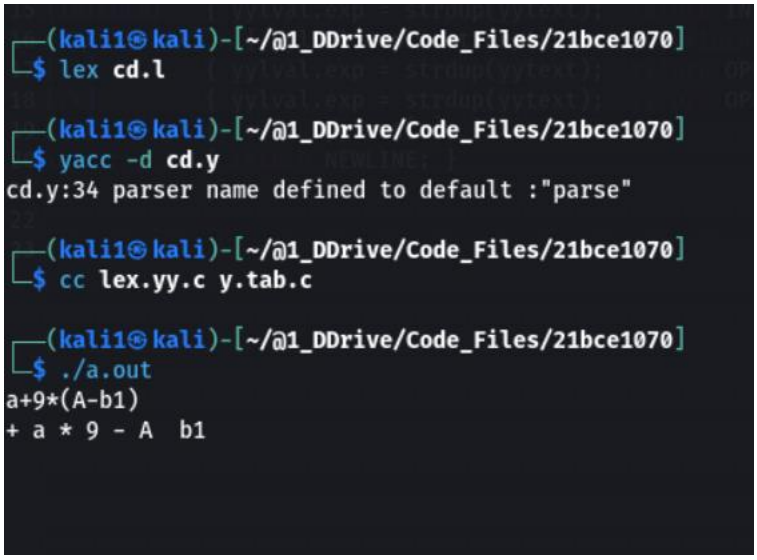
.y

```
%{

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

extern int yylex();
int yyerror(const char *p) { printf("%s\n",p); return 1; }

char *concat( const char* s1, const char* s2, const char*s3)
{
   int len = strlen(s1) + strlen(s2) + strlen(s3) + 1;
   char *s = malloc(sizeof(char)*len);

   int i=0;
   for(int j=0; s1[j]!='\0'; j++)
      s[i++] = s1[j];
   for(int j=0; s2[j]!='\0'; j++)
```

```
        s[i++] = s2[j];
    for(int j=0; s3[j]!='\0'; j++)
        s[i++] = s3[j];

    s[i] = '\0';

    return s;
}
%}


%union
{
    char *exp;
    int val;
};
%token INTEGER IDENTIFIER OPR1 OPR2 NEWLINE EXIT
%left OPR1
%left OPR2

%start lines

%%

lines: /*empty*/
     | lines exp NEWLINE { printf("%s\n ",$<exp>2); }
     ;

exp: exp OPR1 exp { $<exp>$ = concat($<exp>2,$<exp>1,$<exp>3); }
   | exp OPR2 exp { $<exp>$ = concat($<exp>2,$<exp>1,$<exp>3); }
   | '(' exp ')'  { $<exp>$ = $<exp>2; }
```

```
  | INTEGER     { $<exp>$ = concat(" ",$<exp>1," "); }
  | IDENTIFIER   { $<exp>$ = concat(" ",$<exp>1," "); }
  | EXIT { exit(0); }
  ;

%%

int yywrap()
{

   return 1;
}

int main()
{

   yyparse();
}
```

Output



Infix to Prefix (Numerical Expression)

Code

.l

```
%{

   #include <stdio.h>
   #include <stdlib.h>
   #include <string.h>
   #include "y.tab.h"
%}

INTEGER    [0-9]+
IDENTIFIER [_a-zA-Z][_a-zA-Z0-9]*
%%

exit.*      { return EXIT; }
quit.*      { return EXIT; }
{INTEGER}   { yylval.exp = strdup(yytext);  return INTEGER;}
[+-]        { yylval.exp = strdup(yytext);  return OPR1;  }
[/*]        { yylval.exp = strdup(yytext);  return OPR2;  }
[()]        { return yytext[0]; }
\n          { return NEWLINE; }
.           ;

%%
```

.y

```
%{

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

extern int yylex();
int yyerror(const char *p) { printf("%s\n",p); return 1; }

char *concat( const char* s1, const char* s2, const char*s3)
{
   int len = strlen(s1) + strlen(s2) + strlen(s3) + 1;
   char *s = malloc(sizeof(char)*len);

   int i=0;
   for(int j=0; s1[j]!='\0'; j++)
      s[i++] = s1[j];
   for(int j=0; s2[j]!='\0'; j++)
      s[i++] = s2[j];
   for(int j=0; s3[j]!='\0'; j++)
      s[i++] = s3[j];

   s[i] = '\0';

   return s;
}
%}

%union
{
```

```
   char *exp;
   int val;
};
%token INTEGER IDENTIFIER OPR1 OPR2 NEWLINE EXIT
%left OPR1
%left OPR2

%start lines

%%

lines: /*empty*/
    | lines exp NEWLINE { printf("%s\n ",$<exp>2); }
    ;

exp: exp OPR1 exp { $<exp>$ = concat($<exp>2,$<exp>1,$<exp>3); }
  | exp OPR2 exp { $<exp>$ = concat($<exp>2,$<exp>1,$<exp>3); }
  | '(' exp ')'  { $<exp>$ = $<exp>2; }
  | INTEGER     { $<exp>$ = concat(" ",$<exp>1," "); }
  | EXIT { exit(0); }
  ;

%%

int yywrap()
{

   return 1;
}

int main()
```

```
{

   yyparse();

}
```

Output