LAB PROGRAMS FOR INTERNALS

Ashton Prince Rodrigues

LIST OF LEX PROGRAMS

B-1

```
% {/*Lex program to recognize a valid arithmetic expression using + and * and count the number of
identifiers and operators in the expression*/

int id=0, op=0, flag=0;

%}

%%

[0-9]+  {id++,printf("%s is an identifier\n",yytext);}

[+ *] {op++, printf ("%s is an operator\n”, yytext);}

.  {flag=1;}

\n {return 0;}

%%

int main ()

{

   printf("Please enter a valid arithmetic expression\n");

   yylex ();

   if (flag==0 && id==op+1)

   {

      printf ("Valid arithmetic expression.");

      printf("The number of identifiers are %d\n",id);

      printf ("The number of operators are %d\n”, op);

   }

   else

   {

      printf ("Invalid arithmetic expression! \n");

   }

}
```

B-2

```
%{/*Lex program to eliminate comment lines from C program input file and copy all contents into an output file*/
int count=0;
int yywrap()
{
   return 1;
}
%}
%%
"\"".*"\""  ECHO;
"//".*  {count++;}
"/*".*"*/"  {count++;}
%%
int main(int argc,char **argv)
{
   FILE *f1,*f2;
   if(argc > 1)
   {
      f1 = fopen(argv[1],"r");
      if(!f1)
      {
         printf("File cannot be opened!");
         exit(1);
      }
      yyin = f1;
      f2 = fopen(argv[2],"w");
      if(!f2)
      {
         printf("File cannot be opened!");
         exit(1);
```

```
        }
        yyout = f2;
        yylex();
        printf("\n The number of comment lines are %d",count);
    }
    return 0;
}
```

B-3

```
%{/*Lex program to count the number of tabs,lines,spaces and words in an input file.*/
int scnt=0,wcnt=0,lcnt=0,tcnt=0;
int yywrap()
{
    return 1;
}
%}
%%
[ ] {scnt++;}
[^\t\n]+ {wcnt++;}
[\n]    {lcnt++;}
[\t]    {tcnt++;}
%%
main()
{
    FILE *p;
    char fname[30];
    printf("Enter the filename: \n");
    scanf("%s",fname);
    p =fopen(fname,"r");
    if(p==NULL)
    {
        printf("File does not exist!");
    }
    else
    {
        yyin=p;
        yylex();
        printf("The number of words are: %d\n",wcnt);
        printf("The number of spaces are: %d\n",scnt);
        printf("The number of tabs are: %d\n",tcnt);
        printf("The number of lines are: %d\n",lcnt);
    }
}
```

B-4

```
%{ /*LEX program to search for a word in a file - Input word*/
int flag=0;
char word[20];
%}
%%
[a-zA-Z]+ { if(strcmp(yytext, word)==0)
                        flag=1;
          }
%%
int main(int argc, char **argv)
{
        FILE *p;
        char fname[30];
        strcpy(word, argv[1]);
        printf("Enter filename: ");
        scanf("%s", fname);
```

```
	p = fopen(fname, "r");
	if(p==NULL)
		printf("ERROR: file does not exist\n");
	else
	{
		yyin = p;
		yylex();
	if(flag==1)
		printf("Word %s found\n", word);
	else
		printf("Word %s not found\n", word);
	}
	return 1;
}
```

YACC PROGRAMS

C-1

```
%{
#include "lex.yy.c"
%}
%token NUM
%left '+' '-'
%left '*' '/'
%%
stmt: exp { printf("Value of expression = %d\n",$$); }
;
exp: exp '+' exp { $$=$1+$3; }
| exp '-' exp { $$=$1-$3; }
| exp '*' exp { $$=$1*$3; }
| exp '/' exp {
                        if( $3==0 )
                        {
                                printf("Divide by zero error!\n");
                                yyerror();
                        }
                        else
                                $$=$1/$3;
                }
| '(' exp ')' { $$=$2; }
| NUM { $$=$1; }
;
%%
```

```
main()
{
	printf("Enter expression: ");
	yyparse();
	return;
}
int yyerror()
{
	printf("Invalid expression\n");
	exit(0);
}
%{ /*Lexpgm*/
#include "y.tab.h"
extern int yylval;
%}
%%
[0-9]+ { yylval=atoi(yytext); return NUM; }
[\+\-\*\/()] return( yytext[0] );

\n return(0);
. yyerror();
%%
```

C-2

```
%{
#include "lex.yy.c"
int dig=0,id=0,key=0,op=0,lit=0,par=0,inv=0;
%}
%token DIGIT ID KEY OP LIT PAR INV
%%
input:
DIGIT input { dig++;}
|ID input { id++;}
|KEY input { key++;}
|OP input { op++;}
|LIT input { lit++;}
|PAR input { par++;}
|INV input { inv++;}
|DIGIT { dig++;}
|ID { id++;}
|KEY { key++;}
|OP { op++;}
|LIT { lit++;}
|PAR { par++;}
|INV { inv++;}
;
%%
int main(int argc,char **argv)
{
        FILE *f1=fopen(argv[1],"r");
```

```
         if(!f1)
        {
              printf("File cannot be opened\n");
              exit(0);
        }
        yyin=f1;
        do{
                  yyparse();
        }
       while(!feof(yyin));
       printf("Numbers=%d\n Identifiers=%d\n Keywords=%d\n Operators=%d\n
Literals=%d\n Parenthesis=%d\n Invalid Identifiers=%d\n",dig,id,key,op,lit,par,inv);
      return 1;
}
void yyerror()
{
        printf("Parse error! Message: ");
        exit(0);
}
%{
#include "y.tab.h"
extern int yylval;
%}
%%
[\t] ;
int|char|void|bool|float|main|if|else|for|while|return|include<stdio.h>|printf|scanf {printf("%s-
> Keyword\n", yytext);return KEY;}
[0-9]+|[0-9]*[.][0-9]+ {printf("%s-> Number\n",yytext);yylval=atoi(yytext);return DIGIT;}
```

```
[a-zA-Z][a-zA-Z0-9_]* {printf("%s-> Identifier\n",yytext);return ID;}

[+|-|*|/|=|<|>] {printf("%s-> Operator\n",yytext);return OP;}

"\"“. *"\"" {printf("%s-> Literal\n",yytext);return LIT;}

[(|)] {printf("%s-> Parenthesis\n",yytext);return PAR;}

[0-9]+[a-zA-Z]* {printf("%s-> Invalid Identifier\n",yytext);return INV;}

. ;

%%
```

C-3

```
%{
#include "lex.yy.c"
int n,acnt=0,bcnt=0;
%}
%token A B
%%
s:s1 s2 {
                if(acnt==n && bcnt==1)
                {
                        printf("Valid string\n");
                        exit(0);
                }
                 else{
                         printf("Invalid string\n");
                         exit(0);
                }
         }
;
s1:A s1  {acnt++;}
|
;
s2:B s2 {bcnt++;}
|
;
%%
int main(int argc,char **argv)
```

```
{
      n=atoi(argv[1]);
     printf("a should be %d, ending with b",n);
     yyparse();


}
void yyerror()
{
    printf("Invalid\n");
    exit(0);
}
%{
#include "y.tab.h"
%}
%%
a return A;
b return B;
\n return(0);
. yyerror();
%%
```

C-5

```
#include<stdio.h>
#include<string.h>
int z=0,i=0,j=0,c=0;


char a[16],ac[20],stk[15],act[10];
void check();
void main()
{
	printf("GRAMMER is E->E+T|T \n T->T*F|F \n F->(E) \n F->id\n");
	printf("Enter input string\n");
	scanf("%s",a);
	c=strlen(a);
	strcpy(act,"SHIFT->");
	printf("stack \t input \t action");
	for(i=0; j<c; i++,j++)
	{
		if(a[j]=='i' && a[j+1]=='d')
		{
			stk[i]=a[j];
			stk[i+1]=a[j+1];
			stk[i+2]='\0';
			a[j]=' ';
			a[j+1]=' ';
			printf("\n$%s\t%s$\t%sid",stk,a,act);
			check();
		}
		else
		{
			stk[i]=a[j];
			stk[i+1]='\0';
			a[j]=' ';
			printf("\n$%s\t%s$\t%sSYMBOLS",stk,a,act);
			check();
		}
	}
	if((strcmp(stk,"E")==0))
		printf("\n---------\n SUCCESS!!!!!!!!!!\n");
	else
		printf("\n-----------------\nERROR!!!!!!\n");
}

void check()
{
	strcpy(ac,"REDUCE "); //dispaly REDUCE
	for(z=0;z<c;z++)
		if(stk[z]=='(' && stk[z+1]=='E' && stk[z+2]==')')
		{
			stk[z]='F';
```

```
                stk[z+1]='\0';
                printf("\n$%s\t%s$\t%s",stk,a,ac);
                i=i-2;
        }
for(z=0;z<c;z++) // if stack holds id
        if(stk[z]=='i' && stk[z+1]=='d')
        {
                stk[z]='F';
                stk[z+1]='\0';
                printf("\n$%s\t%s$\t%s",stk,a,ac);
                j++;
        }
        for(z=0;z<c;z++)
        {
                if(stk[z]=='T' && stk[z+1]=='*' && stk[z+2]=='F')
                {
                        stk[z]='T';
                        stk[z+1]='\0';
                        printf("\n$%s\t%s$\t%s",stk,a,ac);
                        i=i-2;
                }
                else if(stk[z]=='F')
                {
                        stk[z]='T';
                        printf("\n$%s\t%s$\t%s",stk,a,ac);
                }
        }
        for(z=0;z<c;z++) //checks for stack E+T*
        {
                if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='T'&& stk[z+3]=='*')
                        break;
                if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='T'&& a[j+1]=='*')
                        break;
                else if(stk[z]=='E' && stk[z+1]=='+' && stk[z+2]=='T')
                {
                        stk[z]='E';
                        stk[z+1]='\0';
                        printf("\n$%s\t%s$\t%s",stk,a,ac);
                        i=i-2;
                        return;
                }
        }
        for(z=0;z<c;z++)
        {
                if(stk[z]=='T')
                        if (a[j+1]=='*')
                                break;
                else if(stk[z+1]=='*')
                        break;
                else
                {
                    stk[z]='E';
                    printf("\n$%s\t%s$\t%s",stk,a,ac);
               }
          }
}
```