UNIX SHELL PROGRAMMING AND SYSTEM SOFTWARE LAB

PART A

1. **Write a script to backup list of files**.

```
if [ $# -gt 0 ]
then
        echo "backup files..."
        tar -czvf newbackup.tar $1
else
        echo "please Enter arguments"
fi
```

2. **Write a script that finds all soft links to a specific file.**

```
echo "Enter filename"
read fname
if [ -f $fname ]
then
        echo "Soft links are: "
        ls -l | grep "^l.*.$fname" | cut -d " " -f12
else
        echo "File doesnot exist"
fi
```

3. **Create a script that simulates the ls -l command but print only 3 columns of ur choice**

```
if [ $# -lt 3 ]
then
        echo "enter 3 columns of your choice"
elif [ $1 -le 9 -a $2 -le 9 -a $3 -le 9 ]
then
        echo "The contents of $1,$2,$3:"
        ls -l | cut -d " " -f $1,$2,$3
else
        echo "Please enter column no 1 to 9"
fi
```

4. **Create a script that finds each line in a file that contains a specified string.**

```
echo "Enter string to be search"
read str
echo "enter filename"
read fname
if [ -f $fname ]
then
        echo "The lines containing $str in  $fname:"
        grep $str $fname
else
        echo "File doesnot exist"
fi
```

PART B

1. **A. program to count the number of characters, words,spaces and lines in a given input line.**

```
#include<stdio.h>
int charcount=0;
int wordcount=0;
int linecount=0;
int blankcount=0;
%}

word [^ \t\n]+
eol [\n]
%%
{word}      {wordcount++;charcount+=yyleng;}
{eol}       {linecount++;}
[ ]  {blankcount++;charcount++;}
[\t] {blankcount+=6;charcount+=6;}
%%
int main(int argc,char **argv)
{
    if(argc>1)
    {
            FILE *file;
            file=fopen(argv[1],"r");
            if(!file)
            {
                    printf("Could not open file %s\n",argv[1]);
                    exit(1);
            }
            yyin=file;
            yylex();
            printf("The number of characters:%d\n",charcount);
            printf("The number of word count:%d\n",wordcount);
            printf("The number of blank count:%d\n",blankcount);
            printf("The number of line count:%d\n",linecount);
    }
    else
    {
            printf("Enter the filename along with the program\n");
    }
    return 0;
}
```

B. Program to count no of comment lines in given c program. Also eliminate them and copy the resulting program into separate file.

```
#include<stdio.h>
int cc=0;
%}
%x CMNT
%%
"/*"            {BEGIN CMNT;cc++;}
<CMNT>.                 ;
<CMNT>\n        ;
<CMNT>"*/"      {BEGIN 0;}
%%

int main(int argc, char *argv[])
{
        if(argc!=3)
        {
                printf("Usage: %s <src> <dest> \n",argv[0]);
                return 0;
        }
        yyin=fopen(argv[1],"r");
        yyout=fopen(argv[2],"w");
        yylex();
        printf("No of comment lines %d\n",cc);
        return 0;
}
```

2. A. Program to recognize a valid arithmetic expression and to recognize the identifiers and operators present. print them separately.

```
%{
#include<stdio.h>
int a[]={0,0,0,0},i=0,valid=1,opnd=0;
void ext();
%}
%x OPER
%%
[a-zA-Z0-9]+ {BEGIN OPER;opnd++;}
<OPER>"+" {if(valid){valid=0;i=0;}else ext();}
<OPER>"-" {if(valid){valid=0;i=1;}else ext();}
<OPER>"*" {if(valid){valid=0;i=2;}else ext();}
<OPER>"/" {if(valid){valid=0;i=3;}else ext();}
<OPER>[a-zA-Z0-9]+ {opnd++;if(valid==0){valid=1;a[i]++;}else ext();}
<OPER>"\n" {if(valid==0)ext();else return 0;}
.\n ext();
%%
void ext()
{
    printf("Invalid Expression\n");
    exit(0);
}
int main()
{
    printf("Type the arithmetic expression\n");
    yylex();
    printf("Valid arithemtic expression\n");
    printf("No of operands/indentifiers:%d\n",opnd);
    printf("No of addition:%d\n",a[0]);
    printf("No. of substraction:%d\n",a[1]);
    printf("No of multiplication:%d\n",a[2]);
    printf("No. of division:%d\n",a[3]);
    return 0;
}
```

B. Program to recognize weather a given sentence is a simple or compound.

```
%{
#include<stdio.h>
int f=0;
%}
ws [ \n\t]+
%%
{ws}[aA][nN][Dd]{ws}|{ws}[Oo][Rr]{ws}|{ws}[Bb][Uu][Tt]{ws} f=1;
{ws}[Bb][Ee][Cc][Aa][Uu][Ss][Ee]{ws} f=1;
{ws}[Nn][Ee][Vv][Ee][Rr][Tt][Hh][Ee][Ll][Ee][Ss]{ws} f=1;
. ;
\n return 0;
%%
int main()
{
    printf("Enter a sentence to end press ctrl+d\n");
    yylex();
    if(f==1)
    {
            printf("COMPOUND SENTENCE\n");
    }
    else
    {
            printf("SIMPLE SENTENCE\n");
    }
    return 0;
}
```

3. A. program to recognize a valid arithmetic expression that uses operators '+', '-', '*', '/'.

```
%{
/* lex program */
#include"y.tab.h"
#include<stdlib.h>
int yyerror();
%}
%%
[0-9]+        {return NUM;}
[a-zA-Z][a-zA-Z0-9]*          {return ID;}
[\t] ;
\n   return 0;
. return yytext[0];
%%

%{
/* yacc program */
#include<stdio.h>
#include<stdlib.h>
int yyerror();
%}
%token NUM ID
%left '+' '-'
%left '*' '/'
%%
e:e'+'e | e'-'e | e'*'e | e'/'e | '('e')' | NUM | ID ;
%%
int main()
{
     printf("Enter an expression\n");
     yyparse();
     printf("Valid Expression\n");
     return 0;
}
int yyerror()
{
     printf("Invalid Expression..!!!!\n");
     exit(0);
     return 0;
}
```

B. Program to recognize a valid variable, symbol which starts with a letter followed by any number of letters or digits.

```
/*lex program */
%{
#include"y.tab.h"
%}
%%
[0-9]          {return DIGIT;}
[a-z]          {return LETTER;}
.    {return yytext[0];}
[\n]           {return '\n';}
%%

/*yacc program */
%{
#include<stdio.h>
#include<stdlib.h>
int yylex();
int yyerror();
%}
%token LETTER DIGIT
%%
stmt:var '\n'          {printf("The entered identifier is valid\n"); exit(0);}
|
var:var LETTER
|var DIGIT
|
;
%%
int main()
{
     printf("Enter an identifier\n");
     yyparse();
     return 0;
}
int yyerror()
{
     printf("Invalid identifier\n");
     exit(0);
}
```

4. A. Program to evaluate an arithmetic expression involving operators '+', '-', '*', '/'.

```
/* lex program */
%{
#include<math.h>
#include"y.tab.h"
extern int yylval;
%}
%%
[0-9]   {yylval=atoi(yytext); return NUM;}
[\t]+   ;
[\n]    {return '\n';}
.       {return yytext[0];}
%%

/* yacc  program */
%{
#include<stdio.h>
#include<stdlib.h>
int yyerror();
%}
%token NUM
%left '+' '-'
%left '*' '/'
%%
expr:e  {printf("valid expression \n");
         printf("result %d\n",$$);exit(0);}
e:e'+'e {$$=$1+$3;} |
e'-'e   {$$=$1-$3;} |
e'*'e   {$$=$1*$3;} |
e'/'e   {if($3==0) yyerror("divide by zero error\n");
else    $$=$1/$3;} |
'('e')' {$$=$2;} |
NUM     {$$=$1;} ;
%%
int main()
{
         printf("Enter arithmetic expression:\n");
         yyparse();
}
int yyerror()
{
         printf("invalid expression\n");
         exit(0);
         return 0;
}
```

B. Program to recognize strings 'aaab','abbb','ab','a' using the grammar ($a^n b^m$, n>=1,m>=0).

```
/* lex program */
%{
#include<math.h>
#include<stdlib.h>
#include"y.tab.h"
%}
%%
[a] return A;
[b] return B;
.|\n          return yytext[0];
%%

/*yacc program */
%{
#include<stdio.h>
#include<stdlib.h>
void yyerror();
%}
%token A B
%%
str: RecA RecB '\n'{printf("valid string\n"); exit(0);} ;
RecA: A RecA | A ;
RecB: B RecB | B;
%%
int main()
{
    printf("Enter the string:\n");
    yyparse();
    return 0;
}
void yyerror()
{
    printf("invalid string\n");
    exit(0);
}
```