**SECTION:A**

**UNIX : SHELL**

**PRACTICAL NUMBER 1**

**Write a shell script which accepts any number of arguments from command line and print them in the reverse order ( for example , if script is named rags then executing rags A B C should produce C B A on the standard output ). (Use for loop )**

*Solution:*

clear

echo "THE REVERSE ORDER"

if [ $# -eq 0 ]

then

echo " No arguments "

exit

fi

for i in $\*

do

y=$i" "$y

done

echo $y

*Execution:*

[u110@linux6 shell]$ sh p1 a b c

*Output:*

THE REVERSE ORDER

c b a

**PRACTICAL NUMBER 3**

**Write a Shell program to enhance the inbuilt cal program as below:**

1. **Recognize the month by name e.g. Jan , JAN etc**
2. **Given zero arguments , should print the current month’s calendar**
3. **Given one argument , prints the month or year’s calendar**
4. **Given two arguments, should behave like cal , except for converting month names into numbers.**

*Solution:*

clear

THE REVERSE ORDER

c b a

[u110@linux6 shell]$ cat p3

clear

echo "--------"

echo "CALENDER"

echo "--------"

case $# in

0) set `date`; m=$2; y=$6;;

1) m=$1; set `date`; y=$6;;

\*) m=$1; y=$2;;

esac

case $m in

JAN\*|jan\*|Jan\*) m=1;;

FEB\*|feb\*|Feb\*) m=2;;

MAR\*|mar\*|Mar\*) m=3;;

APR\*|apr\*|Apr\*) m=4;;

MAY\*|may\*|May\*) m=5;;

JUN\*|jun\*|Jun\*) m=6;;

JUL\*|jul\*|Jul\*) m=7;;

AUG\*|aug\*|Aug\*) m=8;;

SEP\*|sep\*|Sep\*) m=9;;

OCT\*|oct\*|Oct\*) m=10;;

NOV\*|nov\*|Nov\*) m=11;;

DEC\*|dec\*|Dec\*) m=12;;

[1-9]|10|11|12) ;;

\*) y=$m; m="";;

esac

cal $m $y

*Execution:*

[u110@linux6 shell]$ sh p3

[u110@linux6 shell]$ sh p3 Aug 16

*Output:*

--------

CALENDER

--------

August 2015

Su Mo Tu We Th Fr Sa

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31

--------

CALENDER

--------

August 16

Su Mo Tu We Th Fr Sa

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31

**PRACTICAL NUMBER 4**

**Write a menu driven shell script to generate the following choices for user :**

1. **To display the file**
2. **To display the permissions of the file**
3. **To find the pattern in the file ignoring the case and case sensitive (using grep)**
4. **To replace all letters ‘e’ by ‘a’**

*Solution :*

clear

echo "MENU"

echo "1. DISPLAY FILE CONTENTS"

echo "2. DISPLAY FILE PERMISSSIONS"

echo "3. FIND PATTERN IN FILE(IGNORE CASE)"

echo "4. REPLACE CHAR E BY A"

read choice

echo "Enter filename"

read file

case $choice in

1)if [ -f $file ]

then

cat $file

else

echo "$file does not exist"

fi;;

2)if [ -f $file ]

then

ls -l $file | cut -c 1-10

else

echo "$file not found"

fi ;;

3) if [ -f $file ]

then

echo " 1) case sensitive 2) case insensitive "

read ch

echo " Enter pattern"

read pattern

case $ch in

1) grep $pattern $file;;

2) grep -i $pattern $file ;;

esac

else

echo "File not found"

fi ;;

4) if [ -f $file ]

then

tr e a < $file

else

echo " $file not found "

fi

;;

esac

*Execution :*

sh p4

*text file contents:*

rbreyfvhdsbf

*file p1 contents:*

clear

echo "THE REVERSE ORDER"

if [ $# -eq 0 ]

then

echo " No arguments "

exit

……

*Output:*

1)

MENU

1. DISPLAY FILE CONTENTS

2. DISPLAY FILE PERMISSSIONS

3. FIND PATTERN IN FILE(IGNORE CASE)

4. REPLACE CHAR E BY A

1

Enter filename

text

rbreyfvhdsbf

2)

MENU

1. DISPLAY FILE CONTENTS

2. DISPLAY FILE PERMISSSIONS

3. FIND PATTERN IN FILE(IGNORE CASE)

4. REPLACE CHAR E BY A

3

Enter filename

p1

1) case sensitive 2) case insensitive

2

Enter pattern

reverse

echo "THE REVERSE ORDER"

**PRACTICAL NUMBER 5:**

**Write a menu driven shell script to generate the following choices for user :**

1. **To display the last n (entered by user) lines from the file**
2. **To sort the file in either ascending order or descending order ( using sort command)**

*Solution:*

clear

echo " MENU "

echo " a) Display last n lines from FILE"

echo " b) Sort the FILE in order"

echo " Enter your choice"

read choice

echo " Enter filename:"

read file

case $choice in

a) if [ -f $file ]

then

echo "Enter the value of n="

read n

n=` expr $n + 1 `

tail -$n $file

else

echo " $file does not exist "

fi;;

b) if [ -f $file ]

then

echo " 1) Ascending Order "

echo " 2) Descending Order "

read ch

case $ch in

1) sort -d $file;;

2) sort -r $file;;

esac

else

echo " $file does not exist "

fi;;

esac

*Execution:*

*The contents of text2:*

hbahsm,ll

090000

-=mmjaansjs

xnmsls.aaoa

mkxkxmaa.cfdgfhfvjhg

dsxzczxv

mn hjyhk

uiouasd

*Output:*

MENU

a) Display last n lines from FILE

b) Sort the FILE in order

Enter your choice

a

Enter filename:

text2

Enter the value of n=

4

dsxzczxv

mn hjyhk

uiouasd

2)

MENU

a) Display last n lines from FILE

b) Sort the FILE in order

Enter your choice

b

Enter filename:

text2

1) Ascending Order

2) Descending Order

2

xnmsls.aaoa

uiouasd

mn hjyhk

-=mmjaansjs

mkxkxmaa.cfdgfhfvjhg

hbahsm,ll

dsxzczxv

090000

**PRACTICAL NUMBER 6**

**Write a shell script to print the Good Morning , Good Afternoon , Good Evening or Good Night according to the time of the day :**

**00:00 AM – 11:59 AM : Good Morning**

**12:00 PM – 3:59 PM : Good Afternoon**

**4:00 PM – 7:59 PM : Good Evening**

**8:00 PM – 11:59 PM : Good Night**

*Solution :*

clear

echo "The Greetings message"

echo `date`

h=`date | cut -c 12-13`

echo $h

if [ $h -lt 12 ]

then echo "Good Morning"

elif [ $h -ge 12 -a $h -lt 16 ]

then echo "Good Afternoon"

elif [ $h -ge 16 -a $h -lt 20 ]

then echo "Good evening"

else

echo "Good Night"

fi

*Execution :*

sh p6

*Output:*

The Greetings message

Sat Aug 29 11:39:29 IST 2015

11

Good Morning

**PRACTICAL NUMBER 7:**

**Write a shell script to list the users currently using the system along with a count of the numbers of times they have logged in .**

*Solution :*

echo " COUNT USER ID"

who | cut -c 1-8 | sort | uniq -c

*Execution :*

sh p7

*Output:*

COUNT USER ID

1 u102

1 u104

1 u105

1 u106

1 u107

1 u109

1 u110

1 u112

1 u113

1 u115

1 u116

1 u117

1 u118

1 u119

**PRACTICAL NUMBER 8**

**Write a shell Program to accept filename or Directory name from the user and only if the particular file exists and not a directory ,allow the user to either**

* 1. **Overwrite the contents of that file**
  2. **append the contents in the previous contents of that file**

*Solution:*

echo " Enter file name or a directory name:"

read nam

if [-d $nam ]

then

echo "This is a directory "

else

echo " This is not a directory "

if [ -f $nam ]

then

echo "The file exist "

echo "a) Do you want to overwrite data ?"

echo "b) Do you want to append the data ?"

read choice

echo "Enter another filename:"

read text

case $choice in

a) cat $nam < $text ;;

b) cat $nam << $text ;;

esac

else

echo "No such file "

fi

fi

*Execution:*

[u110@linux6 shell]$ cat text

rbreyfvhdsbsjfs

rbreyfvhdsbsjfs

[u110@linux6 shell]$ cat text2

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

[u110@linux6 shell]$ sh p8

*Output:*

Enter file name or a directory name:

text

This is not a directory

The file exist

a) Do you want to overwrite data ?

b) Do you want to append the data ?

a

Enter another filename:

text2

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

2)Enter file name or a directory name:

text

This is not a directory

The file exist

a) Do you want to overwrite data ?

b) Do you want to append the data ?

b

Enter another filename:

text2

reyfvhdsbsjfs

reyufhnashdhf

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

**PRACTICAL NUMBER:9**

**Write the shell script to compare two given files, if the contents are same remove the second one.**

*Solution:*

echo "Enter the name of file1 :"

read f1

echo "Enter the name of file2 :"

read f2

if [ -f $f1 ]

then

echo "File exist "

else

echo "File 1 does not exist "

exit

fi

if [ -f $f2 ]

then

echo "File exist "

else

echo "File 2 does not exist "

exit

fi

if cmp -s $f1 $f2

then

echo "Identical files"

rm $f2

else

echo "Non Identical files"

fi

*Execution:*

[u110@linux6 shell]$ cat text

reyfvhdsbsjfs

reyufhnashdhf

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

[u110@linux6 shell]$ cat textcopy

reyfvhdsbsjfs

reyufhnashdhf

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

[u110@linux6 shell]$ sh p9

*Output:*

Enter the name of file1 :

text

Enter the name of file2 :

textcopy

File exist

File exist

Identical files

**PRACTICAL NUMBER 10**

**Write the shell script to merge the contents of three given files, sort the content contained in them and display the sorted output on the screen page by page .**

*Solution:*

cat $1 $2 $3 | sort > f1

cat f1 | more

*Execution:*

[u110@linux6 shell]$ cat text1

xzjajnsuwkisls

tjtirnwek

,

erwwsef

.lk.kl

12

567fghgfjhgj

awasaddddf

'';';'/;

wewdfc

[u110@linux6 shell]$ cat text3

1

2

3

45

7

78

6

5

4

b

s

z

h

wq

[u110@linux6 shell]$ cat text2

grshytpho

\\\\\\

drfhgio

bf

bdvc

fdg

dfg

breyfvhdsbsjfs

[u110@linux6 shell]$ sh p10 text1 text2 text3

Output:

,

'';';'/;

\\\\\\

1

12

2

3

4

45

5

567fghgfjhgj

6

7

78

awasaddddf

b

bdvc

bf

breyfvhdsbsjfs

dfg

drfhgio

erwwsef

fdg

grshytpho

h

.lk.kl

s

tjtirnwek

wewdfc

wq

xzjajnsuwkisls

z

**PRACTICAL NUMBER 11**

**Write a shell program to modify the inbuilt cal program to be able to handle following input:**

**$ Cal jan mar nov**

*Solution:*

clear

y=`date | cut -c 25-28`

for i in $\*

do

case $i in

Jan|JAN|jan) m=1;;

Feb|FEB|feb) m=2;;

Mar|MAR|mar) m=3;;

Apr|APR|apr) m=4;;

May|MAY|may) m=5;;

Jun|JUN|jun) m=6;;

Jul|JUL|jul) m=7;;

Aug|AUG|aug) m=8;;

Sep|SEP|sep) m=9;;

Oct|OCT|oct) m=10;;

Nov|NOV|nov) m=11;;

Dec|DEC|dec) m=12;;

esac

cal $m $y

done

*Execution:*

sh p11 jan mar nov

*Output*

January 2015

Su Mo Tu We Th Fr Sa

1 2 3

4 5 6 7 8 9 10

11 12 13 14 15 16 17

18 19 20 21 22 23 24

25 26 27 28 29 30 31

March 2015

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30 31

November 2015

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30

**PRACTICAL NUMBER 12**

**Write a Shell program to modify the inbuilt cal program to be able to handle following input:**

**$cal jan – oct**

*Solution:*

clear

y=` date | cut -c 25-28`

x=1

for i in $\*

do

case $i in

jan|JAN|Jan) m=1;;

feb|FEB|Feb) m=2;;

mar|MAR|Mar) m=3;;

apr|APR|Apr) m=4;;

may|MAY|May) m=5;;

jun|JUN|Jun) m=6;;

jul|JUL|Jul) m=7;;

aug|AUG|Aug) m=8;;

sep|SEP|Sep) m=9;;

oct|OCT|Oct) m=10;;

nov|NOV|Nov) m=11;;

dec|DEC|Dec) m=12;;

-);;

esac

if [ $# -eq $x ]

then

m2=$m

else

m1=$m

fi

x=`expr $x + 1 `

echo " $x "

done

while [ $m1 -le $m2 ]

do

m=$m1

cal $m $y

m1=` expr $m1 + 1`

done

*Execution:*

sh p12 mar – aug

*Output:*

March 2015

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30 31

April 2015

Su Mo Tu We Th Fr Sa

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30

May 2015

Su Mo Tu We Th Fr Sa

1 2

3 4 5 6 7 8 9

10 11 12 13 14 15 16

17 18 19 20 21 22 23

24 25 26 27 28 29 30

31

June 2015

Su Mo Tu We Th Fr Sa

1 2 3 4 5 6

7 8 9 10 11 12 13

14 15 16 17 18 19 20

21 22 23 24 25 26 27

28 29 30

July 2015

Su Mo Tu We Th Fr Sa

1 2 3 4

5 6 7 8 9 10 11

12 13 14 15 16 17 18

19 20 21 22 23 24 25

26 27 28 29 30 31

August 2015

Su Mo Tu We Th Fr Sa

1

2 3 4 5 6 7 8

9 10 11 12 13 14 15

16 17 18 19 20 21 22

23 24 25 26 27 28 29

30 31

**UNIX: AWK**

**PRACTICAL NUMBER 13**

**Write an awk script to delete duplicated line from a text file. The order of the original lines must remain unchanged.**

*Solution:*

{

if($0 in a) next;

print;

a[$0]=1;

}

*Execution:*

awk -f p13 p13i

*p13i contents:*

car

doll

apple

car

doll

*Output:*

car

doll

apple

**PRACTICAL NUMBER 14**

**Write an AWK Program to implement any sorting technique (Bubble/Selection/Insertion)**

*Solution:*

BEGIN{n=1}

{a[n]=$1

n=n+1}

END{

for(i=1;i<n;i++)

{

for(j=1;j<=n-1-i;j++)

{

if(a[j]>a[j+1])

{

temp=a[j]

a[j]=a[j+1]

a[j+1]=temp

}

}

}

printf(" \n The sorted array is : \n ");

for(i=0;i<n;i++)

{

print a[i];

printf("\n");

}

}

*Execution:*

[u110@linux6 awk]$ awk -f p14

10

9

1

8

*Output:*

The sorted array is :

1

8

9

10

**PRACTICAL NUMBER 15**

**Write a shell script to check the existence of file in the current directory and folds lines of text of a file beyond 30 characters (using AWK)**

*Solution:*

if [ -f $1 ]

then

echo "Valid File"

echo

awk ' {

line= $0

while( (length(line)) > 30 )

{

printf("%s \n",substr(line,1,30));

line=substr(line,31,length(line)-30);

}

print line;

}' $1

else

echo "File does not exist"

fi

*Execution:*

[u110@linux6 awk]$ sh p15 text

text content:

a new file with folds after completing 30 characters.

This is just a test file with more than 30 characters in each recored.

We are using this file to implement shell using awk commands

*Output:*

Valid File

a new file with folds after co

mpleting 30 characters.

This is just a test file with

more than 30 characters in eac

h recored.

We are using this file to impl

ement shell using awk commands

**PRACTICAL NUMBER 16**

**Write an AWK script that accepts date argument in the form of mm-dd-yy and displays it in the form of day, month, and year .The script should check the validity of the argument and in case of error, display a suitable message.**

*Solution:*

BEGIN{

print ARGV[1]

mm=substr(ARGV[1],1,2)

dd=substr(ARGV[1],4,2)

yy=substr(ARGV[1],7,2)

print mm

print dd

print yy

if(mm>"12" || mm<"01")

print "Invalid Month"

else if(mm=="01")

print "Month is January";

else if(mm=="02")

print "Month is February";

else if(mm=="03")

print "Month is March";

else if(mm=="04")

print "Month is April";

else if(mm=="05")

print "Month is May";

else if(mm=="06")

print "Month is June";

else if(mm=="07")

print "Month is July";

else if(mm=="08")

print "Month is August";

else if(mm=="09")

print "Month is September";

else if(mm=="10")

print "Month is October";

else if(mm=="11")

print "Month is November";

else

print "Month is December";

if(mm=="02")

{

if(yy%4==0)

{

if(dd<="29" && dd>="01")

print "Date is " dd;

else

print "Date is invalid";

}

else

{

if(dd<="28" && dd>="01")

print "Date is " dd;

else

print "Date is invalid";

}

}

else if(mm=="04" || mm=="06" || mm=="09" || mm=="11")

{

if(dd<="30" && dd>="01")

print "Date is " dd

else

print "Invalid";

}

else

{

if(dd<="31" && dd>="01")

print "Date is "dd;

else

print "Invalid"

}

print "Year is "yy ;

}

*Execution:*

awk -f p16 09/31/15

*Output:*

09/31/15

09

31

15

Month is September

Invalid

Year is 15

**PRACTICAL NUMBER 17**

**Write an AWK script that accepts two file names as arguments, check if their permissions are identical (or) Different.**

*Solution:*

BEGIN{

system("ls -l "ARGV[1]"|cut -c 1-10 > p17o1")

system("ls -l "ARGV[2]"|cut -c 1-10 > p17o2")

if(system(" cmp -s p17o1 p17o2 "))

print "Different permissions"

else

print "Same permissions"

}

*Execution:*

[u110@linux6 awk]$ awk -f p17 p13 p14

-rw-rw-r--. 1 u110 u110 39 Sep 5 10:48 p13

-rw-rw-r--. 1 u110 u110 224 Sep 19 09:51 p14

*Output:*

Same permissions

**SECTION:B**

**UNIX: LEX**

**PRACTICAL NUMBER 1**

**Write a Lex Program to count the number of lines and characters in the input file.**

*Solution:*

%option noyywrap

%{

int charcount=0,linecount=0;

%}

%%

. charcount++;

\n {linecount++;charcount++;}

%%

int main()

{

yyin=fopen("myfile.txt","r");

yylex();

printf("There are %d characters in %d lines \n",charcount,linecount);

return 0;

}

*Execution:*

[u110@linux6 lex]$ cat myfile.txt

A file for lex program

This is a test file for checking the working of the lex program.

This can be used for other purposes as well.

See you tomorrow.

bye

[u110@linux6 lex]$ lex b1.l

[u110@linux6 lex]$ gcc -o b1 lex.yy.c

[u110@linux6 lex]$ ./b1

*Output:*

There are 158 characters in 7 lines

**PRACTICAL NUMBER 2**

**Write a Lex Program that implements the Caeser cipher : it replaces every letter with the one three letters after in alphabetical order , wrapping around at Z.**

*Solution:*

%option main

%%

[a-wA-W] printf("%c",yytext[0]+3);

[xyzXYZ] printf("%c",yytext[0]-23);

%%

*Execution:*

[u110@linux6 lex]$ lex b2.l

[u110@linux6 lex]$ gcc -o b2 lex.yy.c

[u110@linux6 lex]$ ./b2

*Output:*

abc

def

deff

ghii

ghi

jkl

**PRACTICAL NUMBER 3**

**Write a Lex Program that finds the longest word (defined as a contigious string of upper and lower case letters) in the input.**

*Solution:*

%option noyywrap

%{

#include<string.h>

int longest=0;

char longword[60];

%}

%%

[a-zA-Z]+ {

if(yyleng >longest)

{

longest=yyleng;

strcpy(longword,yytext);

}

}

. ;

\n ;

%%

int main()

{

yylex();

printf("The longest word was \" %s \" which was %d characters long \n",longword,

longest);

return 0;

}

*Execution:*

[u110@linux6 lex]$ lex b3.l

[u110@linux6 lex]$ gcc -o b3 lex.yy.c

[u110@linux6 lex]$ ./b3

a bb ccc dddd

sad hghj

jhyytghn

*Output:*

The longest word was " jhyytghn " which was 8 characters long

**PRACTICAL NUMBER 4**

**Write a Lex program that distinguishes keywords, integers, floats, identifiers, operators, and comments in any simple programming language.**

*Solution:*

%option noyywrap

%{

#include<stdio.h>

%}

DIGIT [0-9]

ID [a-z][a-z0-9]\*

COMMENT \/\\*.\*\\*\/

%%

{DIGIT}+ {printf("\n An integer :%s \n ",yytext);}

{DIGIT}+"."{DIGIT}+ {printf("\n A Floating point number :%s \n",yytext);}

"void"|"int"|"char" {printf("\n Keyword :%s \n",yytext);}

{ID} {printf("\n Identifier :%s \n ",yytext);}

{COMMENT} {printf("\n Comment :%s \n ",yytext);}

[+|-|\*|\/]

[ \t\n] ;

. ;

%%

main(int argc,char\* argv[])

{

yyin=fopen(argv[1],"r");

yylex();

return 0;

}

*Execution:*

[u110@linux6 lex]$ lex b4.l

[u110@linux6 lex]$ gcc -o b4 lex.yy.c

[u110@linux6 lex]$ ./b4 mainprog.cpp

[u110@linux6 lex]$ cat mainprog.cpp

main()

{

int a,b;

char c;

float d,e,f;

}

*Output:*

Identifier :main

Keyword :int

Identifier :a

Identifier :b

Keyword :char

Identifier :c

Identifier :float

Identifier :d

Identifier :e

Identifier :f

**PRACTICAL NUMBER 5**

**Write a LEX program to count the number of identifiers in a C file.**

*Solution:*

%option noyywrap

%{

#include<stdio.h>

int id\_cnt=0;

char ch;

%}

%%

"int"|"float"|"double"|"char" { ch=input();

for(;;)

{

if(ch==',')

id\_cnt++;

else if(ch==';')

break;

ch=input();

}

id\_cnt++;

}

%%

main(int argc, char\* argv[])

{

yyin=fopen(argv[1],"r");

yylex();

printf(" The Number of identifiers in the program = %d \n", id\_cnt);

}

*Execution:*

[u110@linux6 lex]$ lex b5.l

[u110@linux6 lex]$ gcc -o b5 lex.yy.c

[u110@linux6 lex]$ ./b5 mainprog.cpp

[u110@linux6 lex]$ cat mainprog.cpp

main()

{

int a,b;

char c;

float d,e,f;

}

*Output:*

main()

{

}

The Number of identifiers in the program = 6

**PRACTICAL NUMBER 6**

**Write a LEX program to count the number of words, characters, blank spaces and lines in a C file.**

*Solution:*

%option noyywrap

%{

#include<stdio.h>

int cc=0, bc=0, wc=0, lc=0;

%}

%%

[^ \t\n]+ { wc++;

cc=cc+yyleng;

}

\n lc++;

" " bc++;

\t bc=bc+5;

%%

main(int argc,char\* argv[])

{

yyin=fopen(argv[1],"r");

yylex();

printf(" The number of words = %d \n The number of characters = %d \n ", wc,cc);

printf(" The number of blanks = %d \n The number of lines = %d \n ",bc,lc);

}

*Execution:*

[u110@linux6 lex]$ cat myfile.txt

A file for lex program

This is a test file for checking the working of the lex program.

This can be used for other purposes as well.

See you tomorrow.

bye

[u110@linux6 lex]$ lex b6.l

[u110@linux6 lex]$ gcc -o b6 lex.yy.c

[u110@linux6 lex]$ ./b6 myfile.txt

*Output:*

The number of words = 31

The number of characters = 124

The number of blanks = 27

The number of lines = 7

**PRACTICAL NUMBER 7**

**Write a LEX specification program that generates a C program which takes a string**

**“abcd” and prints the following output.**

**abcd**

**abc**

**ab**

**a**

*Solution:*

%option noyywrap

%{

#include<stdio.h>

%}

%%

a|ab|abc|abcd printf("%s\n",yytext);REJECT;

. ;

\n ;

%%

int main()

{

yylex();

return 0;

}

*Execution:*

[u110@linux6 lex]$

[u110@linux6 lex]$ lex b7.l

[u110@linux6 lex]$ gcc -o b7 lex.yy.c

[u110@linux6 lex]$ ./b7

abcd

*Output:*

abcd

abc

ab

a

**PRACTICAL NUMBER 9:1**

**Write a YACC program to find the validity of a given expression (for operators + - \* and /)**

*Solution:*

***LEX part:***

%option noyywrap

%{

#include "y.tab.h"

%}

%%

[0-9]+ return digit;

[a-zA-Z][a-zA-Z0-9]\* return id;

[+-//\*] return op;

.|[\n] return 0;

%%

***YACC part:***

%{

#include "lex.yy.c"

int valid=1;

%}

%token digit id op

%left op

%%

start : exp ;

exp : exp op exp

|id

|digit

;

%%

int yyerror()

{

printf("\n Invalid Expression ");

valid=0;

return 0;

}

int main()

{

printf( "\n Enter expression = \n");

yyparse();

if(valid)

printf("\n A valid Expression \n ");

}

*Execution:*

[u110@linux6 lex]$ yacc -d b91.y

[u110@linux6 lex]$ lex b91.l

[u110@linux6 lex]$ gcc -o b91 y.tab.c

[u110@linux6 lex]$ ./b91

*Output:*

1. Enter expression =

a=b\*\*\*c;

A valid Expression

1. Enter expression =

a+

Invalid Expression

**PRACTICAL NUMBER 9:2**

**Write a program in YACC which recognizes a valid variable which starts with letter followed by a digit. The letter should be in lower case.**

*Solution:*

***LEX part*:**

%option noyywrap

%{

#include "y.tab.h"

%}

%%

[a-z] return letter;

[0-9] return digit;

.|[\n] return 0;

%%

***YACC part:***

%{

#include "lex.yy.c"

int valid=1;

%}

%token digit letter

%%

start : letter digit

%%

int yyerror()

{

printf("\n Not an Identifier \n");

valid=0;

return 0;

}

int main()

{

printf("\n Enter name to be tested \n ");

yyparse();

if(valid)

printf("\n A valid Identifier \n ");

}

*Execution:*

[u110@linux6 lex]$ yacc -d b92.y

[u110@linux6 lex]$ lex b92.l

[u110@linux6 lex]$ gcc -o b92 y.tab.c

[u110@linux6 lex]$ ./b92

*Output:*

Enter name to be tested

a9

A valid Identifier

**PRACTICAL NUMBER 11**

**Write a program in YACC to recognize the string ‘aaabbb’, ‘aabb’, ‘ab’ of the langauge**

**(anbn , n>=1).**

*Solution:*

***LEX part*:**

%option noyywrap

%{

#include "y.tab.h"

%}

%%

a return A;

b return B;

.|[\n] return 0;

%%

***YACC part:***

%{

#include "lex.yy.c"

int valid=1;

%}

%token A B

%%

start : A start B

|

;

%%

int yyerror()

{

printf("\n Not matched ");

valid=0;

return 0;

}

int main()

{

printf("\n Enter pattern to be matched =");

yyparse();

if(valid)

printf("\n A valid pattern ");

}

*Execution:*

[u110@linux6 lex]$ yacc -d b11.y

[u110@linux6 lex]$ lex b11.l

[u110@linux6 lex]$ gcc -o b11 y.tab.c

[u110@linux6 lex]$ ./b11

*Output:*

1. Enter pattern to be matched =aabb

A valid pattern

1. Enter pattern to be matched =abb

Not matched

**PRACTICAL NUMBER 12**

**Write a program in YACC to recognize the langauge (anb, n>=5). (Output to say input is valid or not)**

*Solution:*

***LEX part*:**

%option noyywrap

%{

#include "y.tab.h"

%}

%%

a return A;

b return B;

.|[\n] return 0;

%%

***YACC part:***

%{

#include "lex.yy.c"

int valid=1;

%}

%token A B

%%

start : A A A A A s B

s : A s

|

;

%%

int yyerror()

{

printf("\n Invalid pattern \n ");

valid=0;

return 0;

}

int main()

{

printf("\n Enter pattern :");

yyparse();

if(valid)

printf("\n Valid Pattern \n ");

}

*Execution:*

[u110@linux6 lex]$ yacc -d b12.y

[u110@linux6 lex]$ lex b12.l

[u110@linux6 lex]$ gcc -o b12 y.tab.c

[u110@linux6 lex]$ ./b12

*Output:*

1. Enter pattern :aaaaab

Valid Pattern

1. Enter pattern :aaab

Invalid pattern