01. Write a LEX program to recognize which is verb which is not.

source code:

```
%{
 #include<stdio.h>
%}
[\t]+;
is |
am|
are|
was|
were|
be |
being|
been|
do|
does|
did |
will |
would|
should|
can |
could|
has|
have|
had |
go {printf("%s ; is a verb\n", yytext);}
[a-zA-Z]+{ printf("%s ; is not a verb\n," yytext );}
.|\n { ECHO;}
%%

int yywrap() { return 1;}
```

```
int main ()
{ printf ("Enter a word: ");
   yylex ();
  return 0;
}
```

output:

Discussion: From this lex program we can find out the verbs. If the word is in the given list it will print " is a verb" otherwise it will print not a verb". From this program we can find out the verbs easily.

02. Write a LEX program to recognize parts of speech of a sentence.

Source code:

```
%{
#include<stdio.h>
%}
%%
[\t]+;
is |
am |
was|
were|
be |
being|
been|
do |
did |
does|
will |
would|
should|
can|
could |
have|
has |
had |
go{printf("%s: is a verb\n", yytext);}
very|
simply|
gently|
quietly|
```

```
calmly |
angrily { printf ("%s : is an adverb\n", yytext);}
to |
from |
behind |
above |
below |
between { printf ("%s: is a preposition \n", yytext);}
if |
then |
and |
but |
or {printf{"%s: is a conjunction \n", yytext);}
their |
my |
your |
his |
her |
its { printf ("%s : is an adjective\n," yytext);}
I |
you |
he |
she |
we |
they | { printf ("%s: is a pronoun \n", yytext);}

[a-zA-Z]+{ printf ("%s : don't recognize, might be noun\n",
                yytext);}

.|\n {ECHO;}
```

```
%%
int yywrap() {return 1;}
int main()
{ printf("Enter a sentence: ");
 yylex();
 return 0;
}
```

output:

Discussion: From this lex program we can find out the parts of speech of every word from a sentence. This program can check each and every word of a sentence which are stored in the program. It will print which word is which parts of speech.

01. Write a Lex program to count the positive numbers negative numbers and functions.

CODE:

```
%{
# include <stdio.h>
# include <string.h>
int
    p=0;
int n=0;
int pf=0;
int nf=0;
%}
DIGIT [0-9]
%%
\+ ? {DIGIT}+ p++
  - {DIGIT}+ n++
\+ ? {DIGIT}* \. {DIGIT}+ pf++
   - {DIGIT}* \. {DIGIT}+ nf++
"\n" printf ("\n positive numbers: % d \n negative numbers:
  % d \n positive numbers in fractions: \d \n Negative numbers
   in fractions: % d ; p, n, pf, nf );
 %%
int yywrap (void) {}

int main()
{ printf ("Enter the Numbers: ");
 yylex();
 return 0;
 }
```

Output:

Discussion: It is a simple Lex program which count the positive number, negative & fractions number. We put the positive value as p, negative value as n, positive fraction as pf, negative fraction as nf. In this code we take some different digits and it count how many positive, negative, fraction numbers are there

02: Write a Lex program to count the number of vowels and consonants in a given string.

CODE:

```
%{
#include<stdio.h>
#include<string.h>
int v= 0
int c=0
%}
%%
[aeiouAEIOU] {v++};
[a-zA-Z] {c++;};
"\n" printf("vowels : %d\n consonants : %d ",v,c);
%%
int yywrap() {}
int main(void)
{ printf("Enter the string : ");
  yylex();
  return 0;
}
```

Output:

Discussion: This is a simple lex program. It count the number of vowels and constants in a given string.

Q3: write a LEX program to count the number of lines, spaces, tabs and characters.

CODE:

```
%{
# include <stdio.h>
# include <stdlib.h>
int l=0, s=0, t=0, ch=0;
%}
%%
[\n] l++
[ ] s++
[ ] t++
[^\n] ch++
%%
int main (void)
{ int yywrap (){}
yylex ();
printf ("\n Lines: %d", l);
printf ("\n Spaces: %d", s);
printf ("\n Tabs : %d", t);
printf ("\n Character: %d", ch);
return 0;
}
int yywrap()
{return(1);}
```

Output:

Discussion: In this program we have to count the number of lines, spaces, tabs and characters in the input. It takes input until we press ctrl z. When we press ctrl z it seem like (^Z) and then we have to press enter to get the proper output.

Q4: Write a Lex program to check whether a number is prime or not.

CODE:

```
%{
#include<stdio.h>
#include<stdlib.h>
int flag c, j;
%}
%%
[0-9]+{c=atoi(yytext);
    if (c== 2)
   {printf ("\n prime number");}
     else if (c==0 ||c==1)
    {printf ("\n Not a prime number");}
  else
   {for (j=2; j<c; j++)
   {if (c% j==0)
     flag=1;
  }
if (flag==1)
printf ("\n Not a prime number");
 else if (flag==0)
printf ("\n prime number:");
     }
}
```

```
%%
int yywrap() {}
int main()
{ printf("Enter the number:");
 yylex();
 return 0;
}
```

Output:

Discussion: In this program we have to find a number is prime or not. We know prime numbers are positive integers having only two factors, 1 and integer itself. where 1 is neither prime not composite. In this program we implement that logic and comple the program.

1. Write a Lex program to check whether a number is even or odd

CODE:

```
%{
#include <stdio.h>
 int i;
 %}
%%
[0-9]+ {i= atoi (yytext);
          if (i%==2)
          printf ("Even");
          else
          printf ("Odd ");}
%%
int yywrap() {
}
int main ()
{
yylex ();
 return 0;
}
```

Output:

Discussion: This is a lex program in where we have to find out whether the ~~it~~ taken input is a even number or a odd number. If we take 20 it will print Even because 20 is an even number whis is divisible by 2 and generates the reminder is 0.

2. Write a lex program to count total number of letter in a sentence.

CODE:

```
%{
#include<stdio.h>
# include <string.h>
  int l=0;
%}
%%
[a-zA-Z] {l++;}
"\n" printf ("No-of letters are: %d ", l);
%%
int yywrap() {}
int main (void) {
yylex();
return 0;
}
```

output:

03. Write a lex program to count words that are less than 10 and greater than 5

CODE:

```
%{
#include <stdio.h>
#include <string.h>
int len=0, counter=0;
%}
%%
[a-zA-Z]+ {len = strlen(yytext);
           if (len<10 && len>5)
           {counter ++;}}
%%
int yywrap (void)
{
return 1;
}
int main()
{ printf ("Enter the string : ");
 yylex();
 printf ("\n%d", counter);
 return 0;
}
```

Output:

Discussion: In this program we have to count words that are less than 10 and greater than 5. This is a conditional program. The program executes only when the length of word of less than 10 and greaten than 5.

01. Write a LEX program for checking a valid URL

Source Code:

```
%{
#include<stdio.h>
%%
((http)|(ftp))s?://[a-zA-Z0-9]{2,}(\.[a-z]{2,3})+
    (\/[a-zA-Z0-9+=?]*)*{printf("\n URL valid \n");}
     +{printf("\n URL Invalid \n");}
%%
int yywrap(void){
return 1;}
void main()
{printf("\n Enter URL: ");
 yylex();
printf("\n");
}
```

Output:

Discussion: We know a valid URL always starts with http, without http the URL will not be said to be valid; In this program we check a valid URL.

02. Write a LEX program to accept a valid integer float number.

Source Code:

```
%{
int valid_int=0, valid_float=0;
%}
%%
^[-+]?[0-9]* valid_int++;
^[-+]?[0-9]*[.][0-9]+$ valid_float++;
%%
int yywrap(void){
return 1;
}
int main(){
yylex();
if (valid_int!=0)
printf("Valid Integer number\n");
else if (valid_float!=0)
printf("Valid Float number\n");
else
printf("Not valid Integer/Float number\n");
return 0;
}
```

Output:

Discassion: In this program we find a valid integer and float number. Integer means which is not a fraction, a whole number. when we take (0-9) number then it will print valid integer number. otherwise it will print valid float number.

03. Write a LEX program to accept string starting with vowel.

CODE:

```
%{
int flag=0;
%}
%%
[aeoiuAEOIU].[a-zA-Z0-9]+ flag=1;
[a-zA-Z0-9]+
%%
int yywrap(void){
return 1;}
int main()
{ yylex();
   if (flag==1)
   printf("Accepted");
 else
      printf("Not Accepted");
}
```

Output:

Discussion: In this program we have to accept the string starting with vowel. If the string start with (a,e,i,o,u,A,E,I,O,U) then it will accept the string & will print Accepted otherwise it will print Not Acceptab

04. Write a LEX program to check valid email.

CODE

```
%{
# include<Stdio.h>
%}
%%
^[a-z][a-z0-9]*(@[A-Za-z]+)(\.[a-z]+)+{printf("val
.*{printf("invalid");}
%%
int yywrap(void){
return 1;}
int main()
{ yylex();
}
```

output:

05. Write a LEX program to check valid Mobile number.

CODE:

```
%{
#include<stdio.h>
%}
%%
[\n] {printf ("\n\n Enter mobile Number: ");}
[0-9][0-9]{10} {printf ("Mobile number valid.");}
.+ {printf ("Mobile number Invalid.");}
%%
int yywrap (void){
return 1;}
int main () {
print f ("\n Enter mobile number: ");
yylex();
return 0;
}
```

Output:

Discussion: We know in our country mobile number has 11 digits. If the mobile number has less than 11 digit the program will print invalid and if the number starts with [0-9] and after that it contain other 10 digits it will print valid.

01. Write a LEX program to implement a simple calculator.

Source code:

```
%{
int op=0, i;
float a,b;
%}
dig [0-9]+([0-9]*)"."([0-9]+)
add "+"
sub "-"
mul "*"
div "/"
pow "^"
In \n
%%
{dig} {digi();}
{add} {op=1;}
{sub} {op=2;}
{mul} {op=3;}
{div} {op=4;}
{pow} {op=5;}
{In} {printf("In The Answer : %f\n\n", a);}
%%
digi()
{ if(op==0)
a=atof(yytext);
switch(op)
{ case1; a=a+b;
  break
 case 2: a=a-b;
  break;
```

```
case 3: a=a*b;
        break;
case 4: a=a/b;
        break;
case 5: *for(i=a; b>1; b--)
        a=a*i;
        break;
}
OP=0;
}

main (int argv, char *argc[])
{ yylex();
}
int yywrap(void){
return 1;
}
```

output:

Discussion: In this code we have made a simple calculation To implement a simple calculator we used switch-case condition in it. we used if-else condition to implement the calculator. Inside the if-else condition we used atof in(yylex) which is used to convert the Ascii input to float. The calculation gives us the result of addition, subtraction, multiplication division & power.