WEEK 2

1. Write a lex program to check whether input is digit or not

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%%
^[0-9]* printf("digit");
^[^0-9]|[0-9]*[a-zA-Z] printf("not a digit");
.;
%%
int yywrap()
{
}
int main()
{
yylex();
return 0;
}
```

OUTPUT

```
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ lex w2p1.l
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ ./a.out
7a
not a digit
7
digit
```

2. Write a lex program to check whether the given number is even or odd.

```
%{
#include<stdio.h>
int i;
%}

%%

[0-9]+     {i=atoi(yytext);
        if(i%2==0)
            printf("Even");
```

```
        else
       printf("Odd");}
%%

int yywrap(){}

int main()
{

   yylex();
   return 0;
}
```

OUTPUT

```
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ lex w2p2.l
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ ./a.out
8
Even
31
Odd
```

3. Write a lex program to check whether a number is Prime or not.

```
%{

  #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()
 {
  yylex();
  return 0;
 }
```

OUTPUT

```
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ lex w2p3.l
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ ./a.out
13

 Prime number
6

 Not a prime number
```

4. Write a lex program to recognize a) identifiers
b) keyword-int and float
c) anything else as invalid tokens.

```
%{

  #include<stdio.h>
%}
alpha[a-zA-Z]
digit[0-9]
%%
```

```
(float|int) {printf("\nkeyword");}
{alpha}({digit}|{alpha})* {printf("\nidentifier");}
{digit}({digit}|{alpha})* {printf("\ninvalid token");}
%%
int yywrap()
{
}
int main()
{
yylex();
return 0;
}
```

OUTPUT

```
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ lex w2p4.l
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ ./a.out
int

keyword
var

identifier
8b

invalid token
```

5. Write a lex program to identify a) identifiers
b) keyword-int and float
c) anything else as invalid tokens

Read these from a text file.

```
%{

  #include<stdio.h>
  char fname[25];
%}
alpha[a-zA-Z]
digit[0-9]
%%
(float|int) {printf("\nkeyword");}
{alpha}({digit}|{alpha})* {printf("\nidentifier");}
{digit}({digit}|{alpha})* {printf("\ninvalid token");}
%%
```

```
int yywrap()
{
}
int main()
{
printf("enter filename");
scanf("%s",fname);
yyin=fopen(fname,"r");
yylex();
return 0;
fclose(yyin);
}
```

OUTPUT

```
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ lex w2p5.l
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ gcc lex.yy.c
bmscecse@bmscecse-OptiPlex-3060:~/Documents/1BM21CS253$ ./a.out
enter filenameinput.txt

keyword
identifier;
```