Q9. Design a DFA in LEX Code which accepts string containing even number of ‘a’ and even number of ‘b’ over input alphabet {a, b}.

ANSWER:

```
%{
%}

%s A DEAD

%%
<INITIAL>a BEGIN A;
<INITIAL>b BEGIN INITIAL;
<INITIAL>[^ab\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("ACCEPTED. \n");}

<A>a BEGIN INITIAL;
<A>b BEGIN A;
<A>[^ab\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("NOT ACCEPTED. \n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("INVALID\n");}
%%

int yywrap()
{
return 1;
}

int main()
{
printf("ENTER STRING\n");
yylex();
return 0;
}
```

OUTPUT:

```
ENTER STRING
aaba
NOT ACCEPTED.
aabb
ACCEPTED.
abababa
ACCEPTED.
aaaa
ACCEPTED.
bbbbbb
ACCEPTED.
aabba
NOT ACCEPTED.
```

Q10. Design a DFA in LEX Code which accepts string containing third last element ‘a’ over input alphabet {a, b}.

ANSWER:

```
%{
%}
%s A B C D E F G DEAD
%%
<INITIAL>a BEGIN A;
<INITIAL>b BEGIN INITIAL;
<INITIAL>[^ab\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>a BEGIN B;
<A>b BEGIN C;
<A>[^ab\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("NOT Accepted\n");}

<B>a BEGIN D;
<B>b BEGIN E;
<B>[^ab\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("NOT Accepted\n");}

<C>a BEGIN F;
<C>b BEGIN G;
<C>[^ab\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<D>a BEGIN D;
<D>b BEGIN E;
<D>[^ab\n] BEGIN DEAD;
<D>\n BEGIN INITIAL; {printf("Accepted\n");}

<E>a BEGIN F;
<E>b BEGIN G;
<E>[^ab\n] BEGIN DEAD;
<E>\n BEGIN INITIAL; {printf("Accepted\n");}
```

```
<F>a BEGIN B;
<F>b BEGIN C;
<F>[^ab\n] BEGIN DEAD;
<F>\n BEGIN INITIAL; {printf("Accepted\n");}

<G>a BEGIN A;
<G>b BEGIN INITIAL;
<G>[^ab\n] BEGIN DEAD;
<G>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%

int yywrap()
{
return 1;
}

int main(){
printf("Enter String: \n");
yylex();
return 0;
}
```

OUTPUT:

```
Enter String:
aaa
Accepted
aab
Accepted
baaa
Accepted
abaa
NOT Accepted
a
NOT Accepted
1
Invalid
baa
NOT Accepted
babbaaa
Accepted
bababaabaa
NOT Accepted
aaabaa
NOT Accepted
```

Q11. Design a DFA in LEX Code to Identify and print Integer & Float Constants and Identifier.

ANSWER:

```
%{
%}

%s A B C DEAD

%%
<INITIAL>[0-9]+ BEGIN A;
<INITIAL>[0-9]+[.][0-9]+ BEGIN B;
<INITIAL>[A-Za-z_][A-Za-z0-9_]* BEGIN C;
<INITIAL>[^\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted. \n");}

<A>[^\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Integer\n");}

<B>[^\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Float\n");}

<C>[^\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Identifier\n");}


<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
 %%

int yywrap(){
 return 1;
}

int main(){
 printf("Enter String: \n");
 yylex();
 return 0;
}
```

OUTPUT:

```
Enter String:
sa
Identifier
1326
Integer
1326.0128
Float
1326.aa01
Invalid
_sssaa
Identifier
g22
Identifier
566sd
Invalid
```

Q12. Design a DFA in LEX Code to accept strings ending in 00.

ANSWER:

```
%{
%}
%s A B DEAD
%%
<INITIAL>0 BEGIN A;
<INITIAL>1 BEGIN INITIAL;
<INITIAL>[^01\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>0 BEGIN B;
<A>1 BEGIN INITIAL;
<A>[^01\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>0 BEGIN B;
<B>1 BEGIN INITIAL;
<B>[^01\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
```

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

int main(){
        printf("Enter String: \n");
        yylex();
        return 0;

}
```

OUTPUT:

```
Enter String:
001
Not Accepted
00
Accepted
010101
Not Accepted
0
Not Accepted
1100
Accepted
11010001010010011
Not Accepted
```

Q13. Design a DFA in LEX Code to accept strings ending with 01.

ANSWER:

```
%{
%}
%s A B DEAD
%%
<INITIAL>0 BEGIN A;
<INITIAL>1 BEGIN INITIAL;
<INITIAL>[^01\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>0 BEGIN A;
<A>1 BEGIN B;
<A>[^01\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>0 BEGIN A;
<B>1 BEGIN INITIAL;
<B>[^01\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
```

```
%%
int yywrap(void)
{
	return 0;
}
int main()
{
	printf("Enter String: \n");
	yylex();
	return 0;

}
```

OUTPUT:


Enter String:
101010101101
Accepted
101010111
Not Accepted
11110011
Not Accepted
1
Not Accepted
0
Not Accepted
01
Accepted
10
Not Accepted

Q14. Design a DFA in LEX Code which accepts strings starting with 11.

ANSWER:

```
%{
%}
%s A B DEAD
%%

<INITIAL>1 BEGIN A;
<INITIAL>[^1\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>1 BEGIN B;
<A>[^1\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>0 BEGIN B;
<B>1 BEGIN B;
<B>[^01\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%
```

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

int main()
{
        printf("Enter String: \n");
        yylex();
        return 0;
}
```

OUTPUT:

```
Enter String:
1101
Accepted
01010
Invalid
11101
Accepted
10101
Invalid
1
Not Accepted
111001
Accepted
00111
Invalid
```

Q15. Design a DFA in LEX Code which accepts strings with odd 1’s and even 0’s.

ANSWER:

```
%{
%}
%s A B C DEAD
%%
<INITIAL>0 BEGIN A;
<INITIAL>1 BEGIN B;
<INITIAL>[^01\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>0 BEGIN INITIAL;
<A>1 BEGIN C;
<A>[^01\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("NOT Accepted\n");}

<B>0 BEGIN C;
<B>1 BEGIN INITIAL;
<B>[^01\n] BEGIN DEAD;
<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<C>0 BEGIN B;
<C>1 BEGIN A;
```

```
<C>[^01\n] BEGIN DEAD;
<C>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}
%%

int yywrap(void)
{
        return 0 ;
}
int main(){
        printf("Enter String: \n");
        yylex();
        return 0;

}
```

OUTPUT:

```
Enter String:
11100
Accepted
01010110
Not Accepted
0110
Not Accepted
0011001
Accepted
010011110
Accepted
1
Accepted
0
NOT Accepted
0
NOT Accepted
```

Q16. Design a DFA in LEX code accept even number of 1’s.

ANSWER:

```
%{
%}

%s A DEAD

%%
<INITIAL>1 BEGIN A;
<INITIAL>0 BEGIN INITIAL;
<INITIAL>[^10\n] BEGIN DEAD;
<INITIAL>\n BEGIN INITIAL; {printf("Accepted\n");}

<A>1 BEGIN INITIAL;
<A>0 BEGIN A;
<A>[^10\n] BEGIN DEAD;
<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;
<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%
```

```
int yywrap()
{
return 1;
}
int main()
{
   printf("Enter String\n");
   yylex();
return 0;
}
```

OUTPUT:

```
Enter String:
101011
Accepted
1100
Accepted
1
NOT Accepted
1010101
Accepted
10
NOT Accepted
0111
NOT Accepted
11010110
NOT Accepted
```