ASSIGNMENT : 1

1 List out phases of a compiles. Write a brief not on Lexical Analyzer

2 Explain Buffer pairs and Sentinels.

3 Find errors and identify the phase of compiler detecting them for following C program segment. Justify your answers.

```
int fi( int);
char a[10], * cptr;
int k = 1 ; int j = 2;
float f;
cptr = a;
if (k);
fi(k);
fi( j )
++k;
*(cptr + 1 ) = 0 ;
++ a;
n + *k ;
```

4 What is a symbol table? Discuss any two data structures suitable for it & compare their merits / demerits. Also compare one pass & two pass compilers.

5 Construct a minimum state DFA for the following RE : (Use subset construction method)
(a/b) * a (a/b) (a/b) #

6 Draw NFA-^ for given RE:
(ab)*a+(a/b) #

7 Draw minimized DFA for following RE:
aa(b/c)*ca+b+a #

8 Draw NFA-^ from transition table. Transform NFA to minimized DFA.

9 Find the RE for following language: (subset of {0,1}*)
a) All string containing at least one 0 and at least one 1.
b) All string containing 0's and 1's. Both are even.
c) All string containing at most one pair of consecutive 1's.
d) All string don't end with 01.

ASSIGNMENT : 2

1	Use Thompson's construction method for following RE: (a/b)*abb(a/b)* #
2	Construct minimized DFA for following RE: (0*1*)*0 #
3	Create NFA-^ using Thompson's construction method and convert it into DFA using subset construction method. ((ab*)/c*)(a/b)*c #
4	Construct NFA for following RE. Show the sequence of moves made by each in processing the input string ababbab. ((^/a)b*)* #

ASSIGNMENT : 3

1 Write an algorithm for eliminating left recursion.

2 Which of the following grammer are ambiguous?Justify your answer.
a) S -> a | Sa | bSS | SSb | SbS
b) S-> a| S+S | SS | S* | (S)
c) S-> S(S)S | ^
d) S-> aS | aSbS | ^
e) S-> SS+ | SS- | a

3 Eliminate left recursion and perform left factoring on given grammer.
a) S -> A
B -> bBc | f
A -> Ad | Ac | aB | ac
b) E -> Ma | Sb
M -> ES | ah
S -> ShE | ^
c) A -> ad | a | ab | abc | b

4 Find first and follow for given grammer.
a) A -> (A)A | ^
b) S -> ACB | cbB | Ba
A -> da | BC
B -> g | ^
C -> h | ^

5 Find Whether the given grammer is LL(1) or not:
a) S -> 1AB | ^
A -> 1AC | 0C
B -> 0S
C -> 1
b) A -> BCx | y
B -> yA | ^
C -> Ay | x

6 Construct a recursive decent parser with backtracking for the following grammer:
S -> aSbS | bSaS | ^
Parse the string with backtracking: aabb$

7 Construct predictive parsing table for following:
S -> A
A -> aB | Ad
B -> bBC | f
C -> g