Problem 1: (Prepositional Logic Graph) [Weight: 4%]

Consider a propositional formula *F* in conjunctive normal form (CNF). Given a truth assignment *T* to the variables of *F*, a k-flip is another assignment *T'*, which disagrees with *T* on at most *k* variables. K-Flip refers to the problem of deciding, given *F*, *T*, and *k*, whether there exists a k-flip which satisfies strictly more clauses of *F* than *T* does. This problem is intractable.

Your task is to develop, implement and asymptotically analyze an algorithm A that takes input of a formula *F* and an integer *k*. The algorithm first generates a random assignment *T* then it improves the assignment using k-flips as long as possible, or until a specified timeout is reached, aiming to satisfy more and more clauses.

Implement and test algorithm A with a basic prototype. Ideally, the prototype should accept input formula *F* in exactly same format as provided in dataset.txt file.

The problem commences with a formula *F* consisting of *n* number of variables which can be used in any sequence. The formula consists of clauses having any number of OR operations on multiple variables and the clauses are joined through AND operations. Suppose there is an initial random assignment *T* of Boolean values to the variables of formula *F*, it is required to find another assignment *T'* that disagrees with *T* on at most *k* number of bits and strictly satisfies a greater number of clauses than the original *T*. For simplicity, the disagreement is represented as *d*. Therefore, *T* and *T'* are identical for $(n - d)$ bits where $1 \leq d \leq k$ exhibits the disagreement constraint.

The details of benchmark dataset are provided in the following table.

Parameter	**Value**
K	{5,10,15,20}
Variables	1040
Clauses	3668

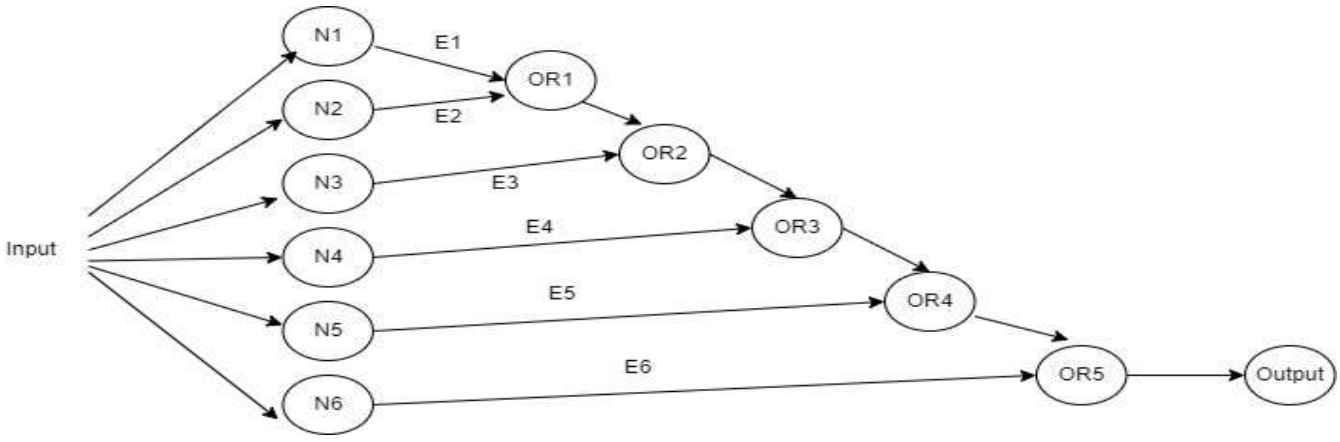

Figure 1 Clause Network

1. Clause Network can have minimum 1 and maximum 6 activated nodes.
2. Each node represents an arbitrary distinct variable.
3. Nodes N2 to N6 are activated only when a clause expression has more than one variable. A sample of Clause Network with edge weights representing a clause {-63, 205, 208, 215, 260} is given below in a table.
4. Weights of E1, E2, E3, E4, E5 and E6 edges represent respective Input Variables.
5. A negative weight edge represents NOT operation on the respective Input Variable.
6. The OR nodes perform OR operation on two values and also have a read logic. For example, -63

7. Use a suitable data structure to represent this sparse graph.
8. Demonstrating a clause-by-clause traversal of the graph is compulsory.
9. Generate node by node graph traversal stats and show these on the output screen.
10. Clauses are provided in a separate dataset.txt file in which each line represents a clause ending with 0. Remember, for testing a different dataset can be used.
11. The initial truth assignment *T* should be randomly generated for all 1040 variables, whereas *T'* is the output of your algorithm that satisfies the given constraint.
12. Provide analysis of your pseudocode, stat files, output and C++ source code.

	N1	N2	N3	N4	N5	N6	OR1	OR2	OR3	OR4	OR5	Output
N1							-63					
N2							205					
N3								208				
N4									215			
N5										260		
N6											0	
OR1								?				
OR2									?			
OR3										?		
OR4											?	
OR5												?
Output												