Experiment #13			
Date		Student ID	
		Student Name	

Experiment Title: Implementation of Programs on NP Hard and NP Complete Problems.

Aim/Objective: To understand the concept and implementation of Basic Programs on NP-Hard and NP-Complete.

Description: The can able to understand the concepts of NP-Hard and NP-Complete

Pre-Requisites:

Knowledge: Non-deterministic Algorithms, 3 SAT Problems, Clique Decision Problems, Node Cover Decision Problems implemented in C/C++/Java/Python.

Tools: Code Blocks/ Eclipse IDE/Python

Pre-Lab: Read the following conversation

Sagar: Is Conjunctive Normal Formula an NP-Hard Problem?

Deepa: Yes, CNF is NP-Hard Problem ?

Sagar: Can you prove...!

You are Deepa's friend. Help her to prove the conjunctive normal formula is a NP-Hard Problem.

- **Procedure/Program**:

1) Define CNF-SAT:

CNF-SAT asks if a CNF formula is satisfiable

2) Show CNF-SAT is in NP: verifying a truth assignment takes Polynomial time

3) Use 3-SAT:

3-SAT is a specific case of CNF-SAT and is NP-Hard

4) Reduction:

Any 3-SAT instance is also a CNF-SAT instance

5) Conclusion:

Since 3-SAT reduces to CNF-SAT, CNF-SAT is NP-Hard.

Experiment #13			
Date		Student ID	
		Student Name	

- **Data and Results:**

CNF-SAT is NP-Hard, reducible from the 3-SAT problem

- **Analysis and Inferences:**

CNF-SAT complexity illustrates fundamental challenges in computational theory

In-Lab:

Given an undirected graph G returns, the minimum number of vertices needed so that every vertex is adjacent to the selected one. In short, return the size of the vertex cover of the graph.

Input:

N=5

M=6

Edges [ ] [ ] = {{1,2}, {4, 1}, {2, 4}, {3, 4}, {5, 2}, {1, 3}}

Output: 3

Explanation: Just pick 2, 3, 4.

- **Procedure/Program:**

```
#include <stdio.h>
#include <stdbool.h>
int vertex_cover(int N, int M, int edges[][2]){
    bool visited[N+1];
    int count=0;
    for(int i=0; i<=N; i++)
```

Course Title	Design and Analysis of Algorithms	ACADEMIC YEAR: 2024-25
Course Code(s)	23CS2205R	Page 84 of 93

```
        visited [i] = false;

for(int i = 0;  i < M; i++) {
  int u = edges[i][0], v = edges[i][1];
if(!visited[u] && !visited[v]) {
visited[u] = visited[v] = true;
      count++;
  }
}
return count;
}
int main() {
int N=5,  M=6;
int edges[][2] = {{1, 2}, {4, 1}, {2, 4}, {3, 4}
                  {5, 2}, {1, 3}};
Printf("%d\n", vertex_cover(N, M, edges));
        return 0;
        }
```

Experiment #13			
Date		Student ID	
		Student Name	

- **Data and Results:**

Data
N=5 edge list : {1,2}, {4,1}, {2,4}, {3,4},
M=6 {5,2} , {1,3}

- **Analysis and Inferences:** Result: vertex cover size is 3

Analysis
selected vertices 2,3,4 cover all graph edges efficiently Inferences
minimum vertex cover ensures efficient edge coverage with fewer vertices

Post-Lab:
Write the procedure to prove that the clique decision problem and Node cover decision problems are NP-Hard Problems.

- **Procedure/Program:**

1) Define Problems:
- clique Decision Problem: find a complete subgraph of size k.

· Node cover Decision Problem: Find a vertex subset of size k covering all edges

2) Show NP membership:

· Both Problems are in NP (verifiable in Polynomial time)

3) Reduction:

· clique to Node cover: in the complement graph G', a clique of size k in G corresponds to a node cover of size $|V| - k$

4) conclusion:

· Both Problems are NP-hard due to Polynomial-time reductions.

Experiment #13			
Date		Student ID	
		Student Name	

- **Data and Results:**

Clique and node cover problems are NP-hard via polynomial-time reductions

- **Analysis and Inferences:**

Clique and node cover problems share similar complexities in graph theory.

- **Sample VIVA-VOCE Questions :**

1. Differentiation between Deterministic and Non-Deterministic Algorithms?
2. What is NP-Complete Problems?
3. Write one word statement about P and NP Class Problems?
4. Mention the relation between P and NP-Hard Problems?
5. Write the Cook's Theorem?

Evaluator Remark (if Any):	**Marks Secured:___out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment

1) Deterministic algorithms produce the same output; non-deterministic algorithms can vary

2) NP-complete problems are NP problems that are the hardset

3) P: solvable in polynomial time;
NP: verifiable in polynomial time.

4) P is a subset of NP-Hard Problems; not vice versa

5) Cook's Theorem: SAT Problem is NP-complete; fundamental in complexity theory