Experiment #13			
Date		Student ID	
		Student Name	

Experiment Title: Implementation of Programs/ Algorithms on CDP, NCDP and AOG problems.

Aim/Objective: To understand the concept and implementation of Basic program on NCDP as NP Hard problem

Description: The students will understand and able to implement programs on NCDP as NP Hard problem.

Pre-Requisites:

Knowledge: NCDP as NP Hard problem in C/C++/PythonTools: Code Blocks/Eclipse IDE

Pre-Lab:

1) Vijval has an array A of length N.

In one operation, Vijval can choose any two **distinct** indices i,j ($1 \le i,j \le N, i=j$) and **either** change Ai to Aj **or** change Aj to Ai.

Find the **minimum** number of operations required to make all the elements of the array **equal**.

Input Format

- First line will contain T, number of test cases. Then the test cases follow.
- First line of each test case consists of an integer N - denoting the size of array A.
- Second line of each test case consists of N space-separated integers 1,2,…,A1,A2,…,AN - denoting thearray A.

Output Format

For each test case, output the minimum number of operations required to make all the elements equal.Constraints

$1 \le T \le 100$

$2 \le N \le 1000$

$1 \le Ai \le 1000$Sample 1:

Input4

4

4 5 6 7

3

6 6 6

4

3 3 2 2

3

2 2 3

Output3

0

2

1

Explanation:

Test Case 1: You can make all the elements equal in 22 operations. In the first operation, you can choose indices 4,5 and convert A1 to A2. So the array becomes [4,4,6,7]. Now you can choose indices 4,6 and convert A6 to A4, etc., so the final array becomes [4,4,4,4].

Test Case 2: Since all the elements are already equal there is no need to perform any operation.

- **Procedure/Program:**

```
#include <stdio.h>
int main(){
int T;
scanf("%d", &T);
while(T--){
int N;
scanf("%d", &N);
int A[N];
int count[1001] = {0};
```

- **Data and Results:**

```
for(int i=0; i<N; i++){
```

```
        scanf("%d", &A[i]);
        count[A[i]]++;
    }

    int maxcount = 0;
    for(int i=1; i<=1000; i++){
        if(count[i] > maxcount){
            maxcount = count[i];
        }
    }
}
int minoperations = N - maxcount;
printf("%d\n", minoperations);
}
return 0;
}
```

Data
=
Test cases illustrate operations needed for equalizing array elements

Result
=
operations required for equal elements in arrays are calculated accurately

Experiment #13			
Date		Student ID	
		Student Name	

- **Analysis and Inferences:**

Analysis

Minimum operations depend on frequency of most common array element

inferences

In-Lab: higher frequency elements reduce required operations for uniform arrays significantly

1. The Clique Decision Problem belongs to NP-Hard. Prove that the Boolean Satisfiability problem reduces to the Clique Decision Problem

- **Procedure/Program:**

1) Input: A CNF formula ϕ with n variables and m clauses

2) graph construction:

- create a graph G with vertices representing literals (both x and $\neg x$)
- connect vertices with edges if:
  - They belong to different clauses
  - They are not contradictory

(i.e, no edges between x and $\neg x$)

3) clique size: set $k = n$

- **Data and Results:**

Conclusion

=

- A satisfying assignment for ϕ corresponds to a clique of size n in G
- Therefore, SAT reduces to the clique Decision Problem, Providing that clique in NP-hard

DATa and Result

$\geq$ $\geq$ $\leq$

SAT reduces to clique Decision Problem, Proving NP - hardness effectively

- **Analysis and Inferences:**

Clique Decision Problem's NP-hardness reflects complexity of Boolean Satisfiability.

Post-Lab

1) You are given two integer A and B.

You need to compute and output the Greatest common divisor and Least common multiple of these 2numbers and store them in the variables GCD and LCM.

Input Format

The first line of input will contain a single integer T, denoting the number of test cases. Each test case consists of a single line of input - the integer A and B.

Output Format

For each test case, output the GCD and LCM on one line

Sample 1:

Input

2

4 9

24 32

Output

1 36

8 96

- **Procedure/Program:**

```
#include<stdio.h>
int gcd(int a, int b){
while(b!=0){
int temp = b;
b = a%b;
a = temp;
```

```
        }
        return a;
    }
int lcm (int a, int b, int gcdvalue)
{
    return (a * b) / gcdvalue;
}
int main(){
    int T;
    scanf ("%d", &T);
    while (T--){
        int A, B;
        scanf ("%d %d", &A, &B);
int gcdvalue = gcd (A, B);
int lcmvalue = lcm (A, B, gcdvalue);
Printf ("%d %d\n", gcdvalue, lcmvalue);
    }
    return 0;
}
```

Experiment #13		Student ID	
Date		Student Name	

- **Data and Results:**

Data Result
2 2 GCD and LCM values
4 9 computed successfully for all
24 32 test cases

- **Analysis and Inferences:**

Analysis
GCD helps simplify fractions; LCM aids in finding common multiples Inferences
GCD and LCM are essential for number theory applications and computations

- **Sample VIVA-VOCE Questions:**

1) Differentiate between CDP and NCDP?
2) List the NP-Hard Graph problems?
3) What is Reducibility?
4) Identify one difference between Satisfiability and Reducibility?
5) What is AOG?

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) CDP is deterministic, NCDP is non-deterministic

Polynomial-time

2) Hamiltonian cycle, clique, vertex cover, graph coloring

3) Process of converting one problem to another efficiently

4) satisfiability checks solutions; reducibility simplifies problem complexity

5) Graph with both "and" and "or" nodes.