Date of the Session:___/___/______ **Time of the Session:_______to________**

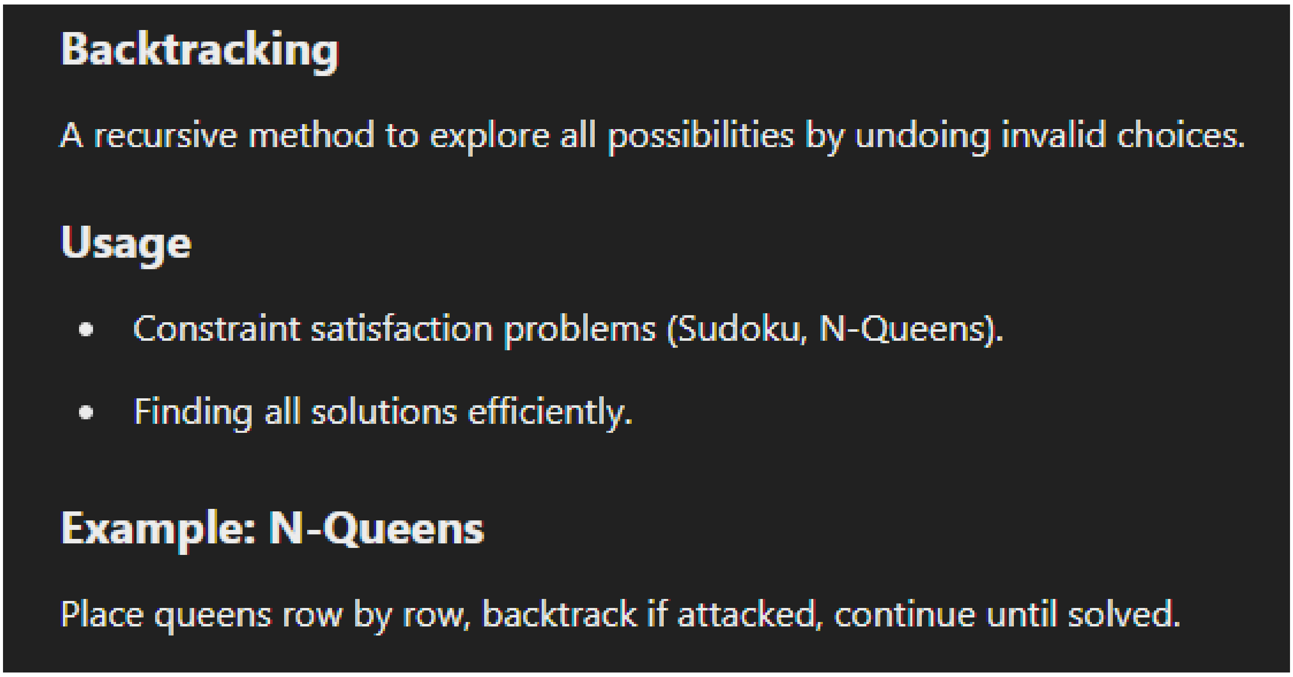
EX – 9 Solving NQueens and Graph Coloring Problems using Back tracking methodology

Prerequisites:

- Basics of Data Structures and C Programming.
- Basic knowledge about Arrays and Graphs.

Pre-Lab:

1) What is Back Tracking methodology and when can we use Back Tracking, explain with an example?



Backtracking

A recursive method to explore all possibilities by undoing invalid choices.

Usage

- Constraint satisfaction problems (Sudoku, N-Queens).
- Finding all solutions efficiently.

Example: N-Queens

Place queens row by row, backtrack if attacked, continue until solved.

2) Mr. Anumula went to the Ice-Cream Parlor. He will be with certain amount of money to buy ice-creams. When he goes to the counter for ordering the Ice-cream, there will be displayed with the items and its cost, respectively. He will be checking which items he can buy according to the money. Print "YES" if he can buy the Ice-Creams without leaving one rupee also otherwise print "NO".

Input

costs= [100, 70, 50, 180, 120, 200, 150]
Total Money=300

Output

YES

```
#include <stdio.h>

int can_buy_ice_creams(int costs[], int n, int total_money) {
    int dp[total_money + 1];
    for (int i = 0; i <= total_money; i++) {
        dp[i] = 0;
    }
    dp[0] = 1;

    for (int i = 0; i < n; i++) {
        for (int j = total_money; j >= costs[i]; j--) {
            if (dp[j - costs[i]] == 1) {
                dp[j] = 1;
            }
        }
    }

    return dp[total_money];
}

int main() {
    int costs[] = {100, 70, 50, 180, 120, 200, 150};
    int total_money = 300;
    int n = sizeof(costs) / sizeof(costs[0]);

    if (can_buy_ice_creams(costs, n, total_money)) {
        printf("YES\n");
    } else {
        printf("NO\n");
    }

    return 0;
}
```

In-Lab:

1) Mani is poor at playing chess, he was asked to arrange “N” Queens on the chess board in such a way that no two queens are allowed to kill each other. Since he is poor at chess you were asked to arrange them on behalf of him. (You must use Backtracking Approach)

Source code:

```
#include <stdio.h>
#include <stdbool.h>

#define N 8

void printSolution(int board[N][N]) {
   for (int i = 0; i < N; i++) {
      for (int j = 0; j < N; j++) {
         printf("%d ", board[i][j]);
      }
      printf("\n");
   }
}

bool isSafe(int board[N][N], int row, int col) {
   for (int i = 0; i < col; i++)
      if (board[row][i])
         return false;

   for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)
      if (board[i][j])
         return false;

   for (int i = row, j = col; i < N && j >= 0; i++, j--)
      if (board[i][j])
         return false;

   return true;
}

bool solveNQueens(int board[N][N], int col) {
   if (col >= N)
      return true;
```

```
     for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1;
            if (solveNQueens(board, col + 1))
                return true;
            board[i][col] = 0;
        }
    }
    return false;
}

void solve() {
    int board[N][N] = {0};
    if (!solveNQueens(board, 0)) {
        printf("Solution does not exist\n");
        return;
    }
    printSolution(board);
}

int main() {
    solve();
    return 0;
}
```

2) Given an undirected graph and N colors, the problem is to find if it is possible to color the graph with at most N colors, which means assigning colors to the vertices of the graph such that no two adjacent vertices of the graph are colored with the same color.
Print "Possible" if it is possible to color the graph as mentioned above, else print "Not Possible".

Input

```
1
3 2
0 2
1 2
2
```

Output

```
Possible
```

Source code:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX_VERTICES 100

bool isSafe(int graph[MAX_VERTICES][MAX_VERTICES], int color[], int vertex, int c, int N) {
   for (int i = 0; i < N; i++) {
      if (graph[vertex][i] == 1 && color[i] == c) {
         return false;
      }
   }
   return true;
}

bool graphColoring(int graph[MAX_VERTICES][MAX_VERTICES], int m, int color[], int vertex,
int N) {
   if (vertex == N) {
      return true;
   }

   for (int c = 1; c <= m; c++) {
      if (isSafe(graph, color, vertex, c, N)) {
         color[vertex] = c;
         if (graphColoring(graph, m, color, vertex + 1, N)) {
            return true;
         }
         color[vertex] = 0;
      }
   }
```

```
    return false;
}

void canColorGraph(int N, int edges[][2], int E, int m) {
    int graph[MAX_VERTICES][MAX_VERTICES] = {0};
    int color[MAX_VERTICES] = {0};

    for (int i = 0; i < E; i++) {
        int u = edges[i][0];
        int v = edges[i][1];
        graph[u][v] = 1;
        graph[v][u] = 1;
    }

    if (graphColoring(graph, m, color, 0, N)) {
        printf("Possible\n");
    } else {
        printf("Not Possible\n");
    }
}

int main() {
    int N = 3;
    int m = 2;
    int edges[][2] = {{0, 2}, {1, 2}};
    int E = sizeof(edges) / sizeof(edges[0]);

    canColorGraph(N, edges, E, m);

    return 0;
}
```

Post-Lab:

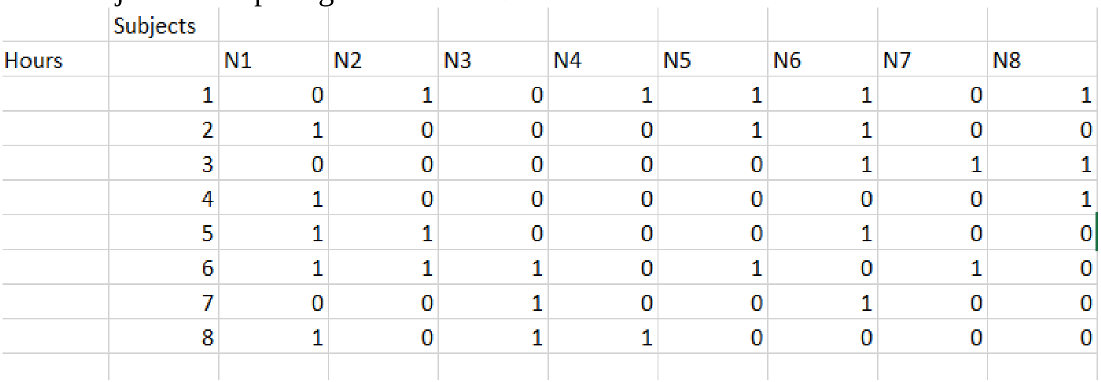
1) John is a very obedient boy and helping others is a habit of him. He will do any work with dedication his teacher assigns him a work to generate all permutations of given word. Since he is busy helping others, you are asked to help him to complete his work on behalf of him. (Use Backtracking Concept)

Source code:

```
#include <stdio.h>
#include <string.h>

void swap(char *x, char *y) {
    char temp;
    temp = *x;
    *x = *y;
    *y = temp;
}

void permute(char *str, int l, int r) {
    if (l == r) {
        printf("%s\n", str);
    } else {
        for (int i = l; i <= r; i++) {
            swap((str + l), (str + i));
            permute(str, l + 1, r);
            swap((str + l), (str + i));
        }
    }
}

int main() {
    char str[] = "ABC";
    int n = strlen(str);
    permute(str, 0, n - 1);
    return 0;
}
```

2) Mr. Sai joined for as an assistant at a school where he was given a job to arrange schedules for the subject n1, n2, n3, n4, n5, n6, n7, n8. Help him schedule the timetable from the given information.

Let this be the schedule:
Here for everyone hour there are many subjects competing. Use backtracking and create a schedule where each subject is assigned to a specific hour in a day. In the schedule '1' represents that the subject is competing for that hour.



	Subjects								
Hours		N1	N2	N3	N4	N5	N6	N7	N8
	1	0	1	0	1	1	1	0	1
	2	1	0	0	0	1	1	0	0
	3	0	0	0	0	0	1	1	1
	4	1	0	0	0	0	0	0	1
	5	1	1	0	0	0	1	0	0
	6	1	1	1	0	1	0	1	0
	7	0	0	1	0	0	1	0	0
	8	1	0	1	1	0	0	0	0

Source code:

```
#include <stdio.h>
#include <stdbool.h>

#define SUBJECTS 8
#define HOURS 8

bool is_safe(int subject, int hour, int schedule[], int conflicts[SUBJECTS][SUBJECTS]) {
  for (int i = 0; i < SUBJECTS; i++) {
    if (schedule[i] == hour && conflicts[subject][i]) {
      return false;
    }
  }
  return true;
}

bool backtrack(int schedule[], int subjects[], int hours[], int conflicts[SUBJECTS][SUBJECTS],
int index) {
  if (index == SUBJECTS) return true;

  for (int i = 0; i < HOURS; i++) {
    if (is_safe(subjects[index], hours[i], schedule, conflicts)) {
      schedule[subjects[index]] = hours[i];
      if (backtrack(schedule, subjects, hours, conflicts, index + 1)) return true;
      schedule[subjects[index]] = -1;
    }
    }
    return false;
}

void schedule_subjects(int conflict_matrix[SUBJECTS][SUBJECTS]) {
    int subjects[SUBJECTS] = {0, 1, 2, 3, 4, 5, 6, 7};
    int hours[HOURS] = {1, 2, 3, 4, 5, 6, 7, 8};
    int schedule[SUBJECTS];
    for (int i = 0; i < SUBJECTS; i++) schedule[i] = -1;

    if (backtrack(schedule, subjects, hours, conflict_matrix, 0)) {
        for (int i = 0; i < SUBJECTS; i++) {
            printf("N%d -> Hour %d\n", i + 1, schedule[i]);
        }
    } else {
        printf("No valid schedule found\n");
    }
}

int main() {
    int conflict_matrix[SUBJECTS][SUBJECTS] = {
        {0, 1, 0, 1, 1, 1, 0, 1},
        {1, 0, 0, 0, 1, 1, 0, 0},
        {0, 0, 0, 0, 0, 1, 1, 1},
        {1, 0, 0, 0, 0, 0, 0, 1},
        {1, 1, 0, 0, 0, 1, 0, 0},
        {1, 1, 1, 0, 1, 0, 1, 0},
        {0, 0, 1, 0, 0, 1, 0, 0},
        {1, 0, 1, 1, 0, 0, 0, 0}
    };

    schedule_subjects(conflict_matrix);
    return 0;
}
```

Comments of the Evaluators (if Any)	Evaluator's Observation Marks Secured:________out of [50]. Signature of the Evaluator Date of Evaluation: