# ASSIGNMENT 4 BACKTRACKING

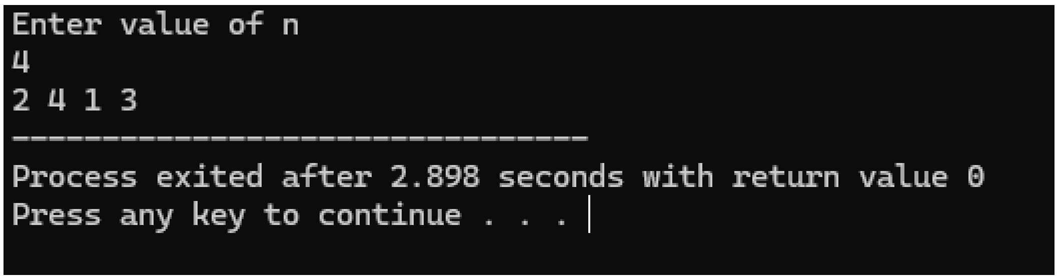
1)
```
#include <bits/stdc++.h>
using namespace std;

int safe(vector<vector<int>>& grid, int m, int n) {
    int size = grid.size();
    int i, j;
    for (i = 0; i < m; i++) {
        if (grid[i][n]) {
            return 0;
        }
    }
    for (i = m - 1, j = n - 1; i >= 0 && j >= 0; i--, j--) {
        if (grid[i][j]) {
            return 0;
        }
    }
    for (i = m - 1, j = n + 1; j < size && i >= 0; i--, j++) {
        if (grid[i][j]) {
            return 0;
        }
    }
    return 1;
}

int place(int m, vector<vector<int>>& grid) {
    int size = grid.size();
    if (m == size) {
        return 1;
    }
    for (int i = 0; i < size; i++) {
        if (safe(grid, m, i)) {
            grid[m][i] = 1;
            if (place(m + 1, grid)) {
                return 1;
            }
            grid[m][i] = 0;
        }
    }
    return 0;
}

vector<int> nQueen(int size) {
    vector<vector<int>> grid(size, vector<int>(size, 0));
    if (place(0, grid)) {
        vector<int> sol;
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                if (grid[i][j]) {
                    sol.push_back(j + 1);
                }
            }
        }
        return sol;
    } else {
        return {-1};
    }
}

int main() {
    int size;
    cout << "Enter value of n:" << endl;
    cin >> size;
    vector<int> sol = nQueen(size);
    for (auto i : sol) {
        cout << i << " ";
    }
    return 0;
}
```



```
Enter value of n
4
2 4 1 3
--------------------------------
Process exited after 2.898 seconds with return value 0
Press any key to continue . . .
```

2)

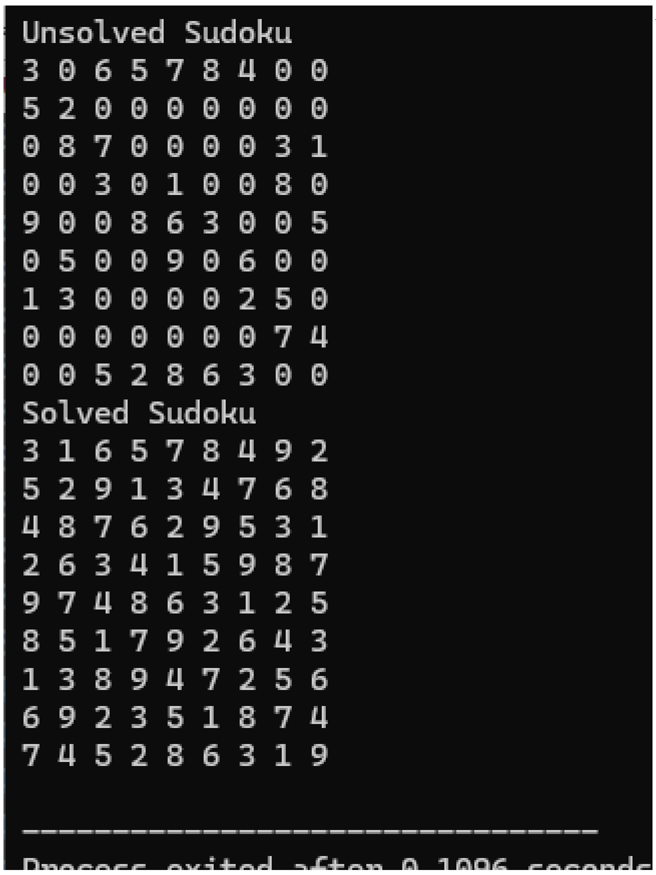
```
#include <iostream>
#include <vector>
using namespace std;

bool safe(vector<vector<int>> &grid, int m, int n, int num) {
    for (int x = 0; x <= 8; x++) {
        if (grid[m][x] == num) {
            return false;
        }
    }
    for (int x = 0; x <= 8; x++) {
        if (grid[x][n] == num) {
            return false;
        }
    }
    int sRow = m - (m % 3), sCol = n - (n % 3);
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (grid[i + sRow][j + sCol] == num) {
                return false;
            }
        }
    }
    return true;
}

bool sudoku(vector<vector<int>> &grid, int m, int n) {
    int size = grid.size();
    if (m == size - 1 && n == size) {
        return true;
    }
    if (n == size) {
        m++;
        n = 0;
    }
    if (grid[m][n] != 0) {
        return sudoku(grid, m, n + 1);
    }
    for (int num = 1; num <= size; num++) {
        if (safe(grid, m, n, num)) {
            grid[m][n] = num;
            if (sudoku(grid, m, n + 1)) {
                return true;
            }
            grid[m][n] = 0;
        }
    }
    return false;
}

void solve(vector<vector<int>> &grid) {
    sudoku(grid, 0, 0);
}

int main() {
    vector<vector<int>> grid = {
        {3, 0, 6, 5, 7, 8, 4, 0, 0},
        {5, 2, 0, 0, 0, 0, 0, 0, 0},
        {0, 8, 7, 0, 0, 0, 0, 3, 1},
        {0, 0, 3, 0, 1, 0, 0, 8, 0},
        {9, 0, 0, 8, 6, 3, 0, 0, 5},
        {0, 5, 0, 0, 9, 0, 6, 0, 0},
        {1, 3, 0, 0, 0, 0, 2, 5, 0},
        {0, 0, 0, 0, 0, 0, 0, 7, 4},
        {0, 0, 5, 2, 8, 6, 3, 0, 0}};
    cout << "Unsolved Sudoku" << endl;
    for (int i = 0; i < grid.size(); i++) {
        for (int j = 0; j < grid.size(); j++) {
            cout << grid[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
    solve(grid);
    cout << "Solved Sudoku" << endl;
    for (int i = 0; i < grid.size(); i++) {
        for (int j = 0; j < grid.size(); j++) {
            cout << grid[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

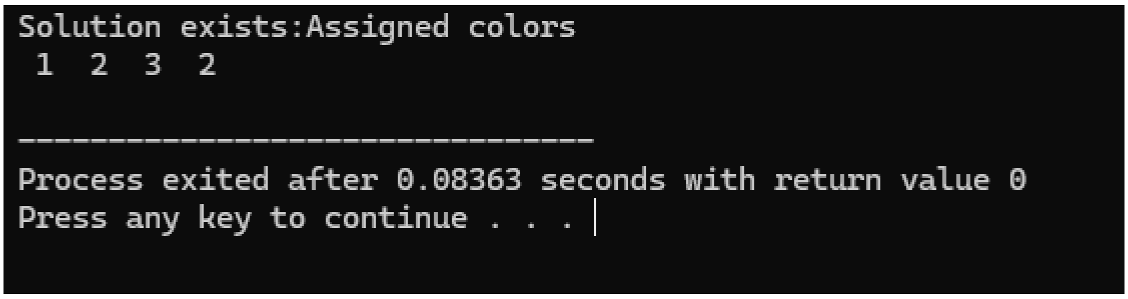


```
Unsolved Sudoku
3 0 6 5 7 8 4 0 0
5 2 0 0 0 0 0 0 0
0 8 7 0 0 0 0 3 1
0 0 3 0 1 0 0 8 0
9 0 0 8 6 3 0 0 5
0 5 0 0 9 0 6 0 0
1 3 0 0 0 0 2 5 0
0 0 0 0 0 0 0 7 4
0 0 5 2 8 6 3 0 0
Solved Sudoku
3 1 6 5 7 8 4 9 2
5 2 9 1 3 4 7 6 8
4 8 7 6 2 9 5 3 1
2 6 3 4 1 5 9 8 7
9 7 4 8 6 3 1 2 5
8 5 1 7 9 2 6 4 3
1 3 8 9 4 7 2 5 6
6 9 2 3 5 1 8 7 4
7 4 5 2 8 6 3 1 9


--------------------------------
Process exited after 0.1096 seconds
```

3)

```
#include <bits/stdc++.h>
using namespace std;
#define V 4
void print(int color[]) {
    cout << "Solution exists:Assigned colors"<<endl;
    for (int i = 0; i < V; i++) {
        cout << " " << color[i] << " ";
    }
    cout << "\n";
}
bool isSafe(int v, bool graph[V][V], int color[], int c) {
    for (int i = 0; i < V; i++) {
        if (graph[v][i] && c == color[i]) {
            return false;
        }
    }
    return true;
}
bool utility(bool graph[V][V], int m, int color[], int v) {
    if (v == V) {
        return true;
    }
    for (int c = 1; c <= m; c++) {
        if (isSafe(v, graph, color, c)) {
            color[v] = c;
            if (utility(graph, m, color, v + 1) == true) {
                return true;
            }
            color[v] = 0;
        }
    }
    return false;
}
bool graphcolor(bool graph[V][V], int m) {
    int color[V];
    for (int i = 0; i < V; i++) {
        color[i] = 0;
    }
    if (utility(graph, m, color, 0) == false) {
        cout << "Solution does not exist";
        return false;
    }
    print(color);
    return true;
}
int main() {
    bool graph[V][V] = {
        { 0, 1, 1, 1 },
        { 1, 0, 1, 0 },
        { 1, 1, 0, 1 },
        { 1, 0, 1, 0 },
    };
    int m = 3;
    graphcolor(graph, m);
    return 0;
}
```



```
Solution exists:Assigned colors
 1  2  3  2

--------------------------------
Process exited after 0.08363 seconds with return value 0
Press any key to continue . . .
```