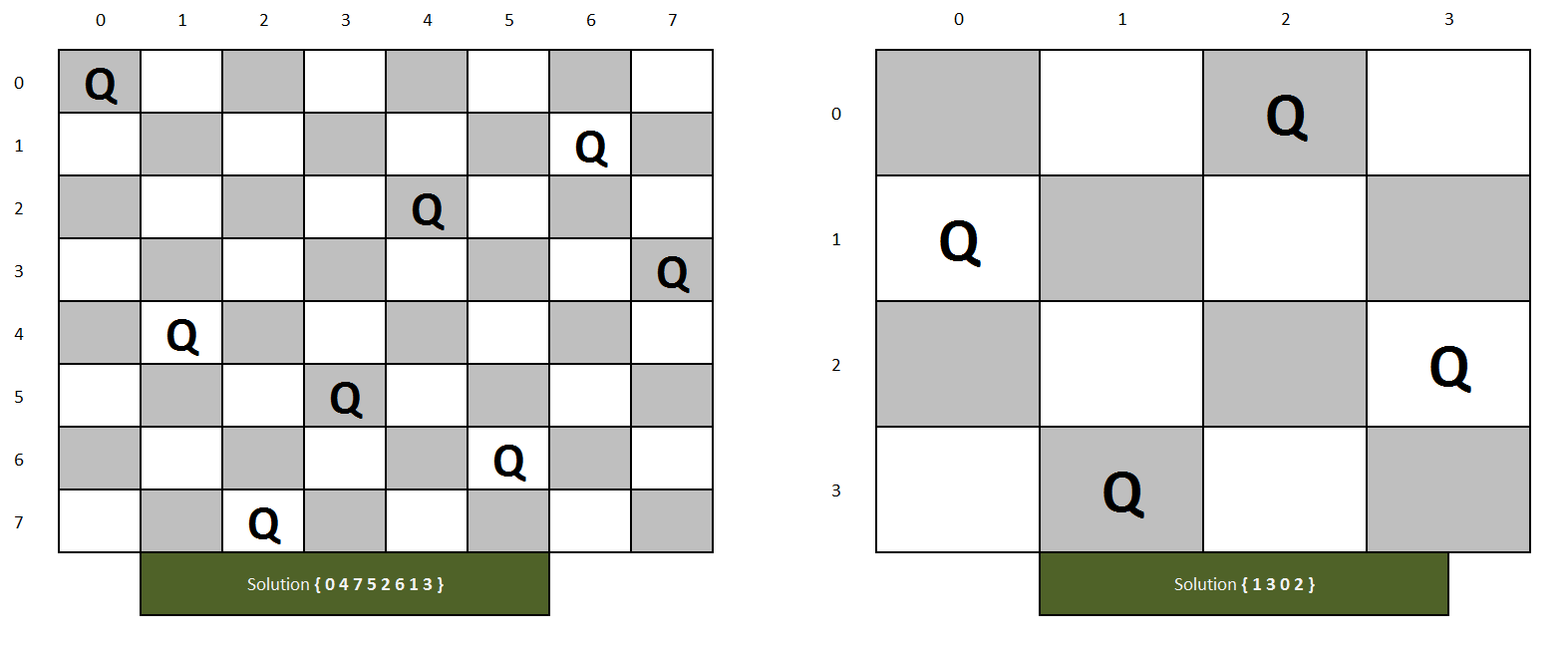
**EXPERIMENT-04**

**OBJECTIVE-** To implement the concept of Artificial Intelligence in N-Queen .

**THEORY OF N-Queen:-**

The n-queens problem consists in placing n non-attacking queens on an n-by-n chess board. A queen can attack another queen vertically, horizontally, or diagonally. E.g. placing a queen on a central square of the board blocks the row and column where it is placed, as well as the two diagonals (rising and falling) at whose intersection the queen was placed.



**ALGORITHM OF N-Queen :-**

1. Place the queens column wise, start from the left most column
2. If all queens are placed.
   1. return true and print the solution matrix.
3. Else
   1. Try all the rows in the current column.
   2. Check if queen can be placed here safely if yes mark the current cell in solution matrix as 1 and try to solve the rest of the problem recursively.
   3. If placing the queen in above step leads to the solution return true.
   4. If placing the queen in above step does not lead to the solution , BACKTRACK, mark the current cell in solution matrix as 0 and return false.
4. If all the rows are tried and nothing worked, return false and print NO SOLUTION.

**CODE IMPLEMENTATION OF N-Queen:-**

#include<iostream>

#include<cmath>

#include<stdlib.h>

using namespace std;

int i,k,j,n;

int x[30];

int count=0;

// for checking the non-attacking condition for placing the queen

bool place(int,int);

void printSolution(int x[])

{

static int count = 0;

count++;

cout<<"#"<<count ;

cout<<endl<<endl;;

for(i = 1; i<=n; i++)

{

for(j= 1; j<=n;j++)

{

if(x[i]== j)

{

cout<<" "<<1;

}

else

{

cout<<" "<< "\_\_\_";

}

}

cout<<endl<<endl;

}

cout<<endl;

}

// Applies the NQueens algorithm for each queen recursively using backtracking

void Nqueen(int k, int n){

// arguement1: k represents the queen number as well as the row to place queen in

// arguement2: n represents the total number of queens

for(int i=1;i<=n;i++)

{

// i is a column number where queen is placed

if(place(k,i))

{

// place queen k in column i

x[k]=i;

if(k==n)

{

cout << "\n solution found!\n";

for(i=1;i<=n;i++){

cout<<"Column number of queen "<<i<<": "<<x[i]<<endl;

}

cout<<"Placement is succesfull"<<endl;

cout<<endl;

printSolution(x);

exit(0);

return;

}

else

{

Nqueen(k+1,n);

}

}

}

}

// Checking if placing a queen at row k and column[row] position on the board is safe

bool place(int k,int i)

{

for(j=1;j<=k-1;j++)

{

// This condition checks if another queen lies on the same column or on the diagonal

cout<<"Checking Position for Queen "<<k<<" in column "<<i<<endl;

if(x[j]==i || abs(x[j]-i)==abs(j-k))

{

cout<<"Column "<<i<<" is not suitable"<<endl<<endl;

return false;

}

}

cout<<"COLUMN "<<i<<" IS SAFE FOR QUEEN:"<< k<<endl<<endl;

return true;

}

int main()

{

cout<<"Enter the number of queens :";

cin>>n;

//Initialize

for(i=0;i<n;i++)

{

x[i]=0;

}

Nqueen(1,n);

}

**OUTPUT: -**

