Practical – 10

Implement N Queen's problem using Backtracking.

```
#include <stdio.h>
#include <stdlib.h>
#define N 8 // The number of queens
int board[N][N]; // The chessboard
int isSafe(int row, int col)
{
   int i, j;

   /* Check the row */
   for (i = 0; i < col; i++)
      if (board[row][i])
         return 0;

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

   for (i = row, j = col; i < N && j >= 0; i++, j--)
      if (board[i][j])
         return 0;
   return 1;
}
int solve(int col)
{
   int row;

   if (col >= N)
      return 1;

   for (row = 0; row < N; row++)
   {
      if (isSafe(row, col))
      {
         /* Place the queen */
         board[row][col] = 1;
         if (solve(col + 1))
            return 1;

         /* Backtrack */
         board[row][col] = 0;
      }
   }

   return 0;
}
```

```
void printSolution()
{
   int i, j;

   for (i = 0; i < N; i++)
   {
      for (j = 0; j < N; j++)
         printf("%d ", board[i][j]);

      printf("\n");
   }
}

int main()
{

   int i, j;
   printf("21012021003_AMIT GOSWAMI \n");
   printf("the solution for %d queens: \n", N);
   for (i = 0; i < N; i++)
      for (j = 0; j < N; j++)
         board[i][j] = 0;

   if (solve(0))
      printSolution();
   else
      printf("No solution found\n");

   return 0;
}
```

```
the solution for 8 queens:
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

```
the solution for 4 queens:
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
```

```
the solution for 3 queens:
No solution found
```