ASSIGNMENT NO. B1

Problem Statement:

8-Queens Matrix is Stored using JSON/XML having first Queen placed, use back-tracking to place remaining Queens to generate final 8-queen's Matrix using Python.

Learning Objectives:

1. To study 8Queen Problem in detail with solution.
2. To study about how to take input from XML.

Learning Outcomes:

Learnt about 8 queens problem and find out solution for it using backtracking method.

Software and Hardware Requirements:

1. 64-bit operating System(Linux)
2. XML File
3. Text editor
4. Terminal
5. Python Compiler
6. Modelio 3.6

THEORY

8Queen Problem:

The eight queens puzzle is the problem of placing eight chess queens on an 8*8 chessboard so that no two queens threaten each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general n-queens problem of placing n queens on an n*n chessboard, where solutions exist for all natural numbers n with the exception of n=2 and n=3.

Backtracking:

Backtracking is a general algorithm for finding all (or some) solutions to some computational problems, notably constraint satisfaction problems, that incrementally builds candidates to the solutions, and abandons each partial candidate c ("backtracks") as soon as it determines that c cannot possibly be completed to a valid solution.

The classic textbook example of the use of backtracking is the eight queens puzzle, that asks for all arrangements of eight chess queens on a standard chessboard so that no queen attacks any other. In the common backtracking approach, the partial candidates are arrangements of k queens in the first k rows of

the board, all in different rows and columns. Any partial solution that contains two mutually attacking queens can be abandoned. Backtracking can be applied only for problems which admit the concept of a "partial candidate solution" and a relatively quick test of whether it can possibly be completed to a valid solution. It is useless, for example, for locating a given value in an unordered table. When it is applicable, however, backtracking is often much faster than brute force enumeration of all complete candidates, since it can eliminate a large number of candidates with a single test. Backtracking is an important tool for solving constraint satisfaction problems, such as crosswords, verbal arithmetic, Sudoku, and many other puzzles.

XML:

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C's XML 1.0 Specification and by several other related specifications, all of which are free open standards. The design goals of XML emphasize simplicity, generality and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures such as those used in web services.

Several schema systems exist to aid in the definition of XML-based languages, while many application programming interfaces (APIs) have been developed to aid the processing of XML data. An XML database is a data persistence software system that allows data to be specified, and sometimes stored, in XML format. These data can be queried, transformed, exported and returned to a calling system. XML databases are a flavor of document-oriented databases which are in turn a category of NoSQL database (meaning Not (only) SQL).

Mathematical Model:

Let S be the solution perspective of the class booths such that
S={s, e, X,Y, F}

s = start state $[A]_{8,8}$
Matrix with some queens placed.

e = end state $[A]_{8,8}$
Matrix with all queens placed.

X = input of the system.
X = $[A]_{8,8}$ where,
For any two queens $a_{i,j}$ and $a_{k,l}$
$0 \leq i,j,k,l \leq 7$
$i \neq k$
$j \neq l$
$| i-k | \neq |j-l|$

F={ solve(),isSafe() }

isSafe(A,i,j):
For any two queens $a_{i,j}$ and $a_{k,l}$

i = k	False
j = l	False
|i-k| = |j-l|	False

solve(A,i,j):

$a_{i,j} = 1$	$0 \leq i \leq 7$
isSafe(A,i,j) = 1	j=j+1
isSafe(A,i,j) = 0	i=i+1
i = 7	$a_{i,j-1} = 0$

Y = { $[A]_{8,8}$ }
where all queens are placed such that
For any two queens $a_{i,j}$ and $a_{k,l}$
$0 \leq i,j,k,l \leq 7$
$i \neq k$
$j \neq l$
$| i-k | \neq |j-l|$

Use Case Diagram

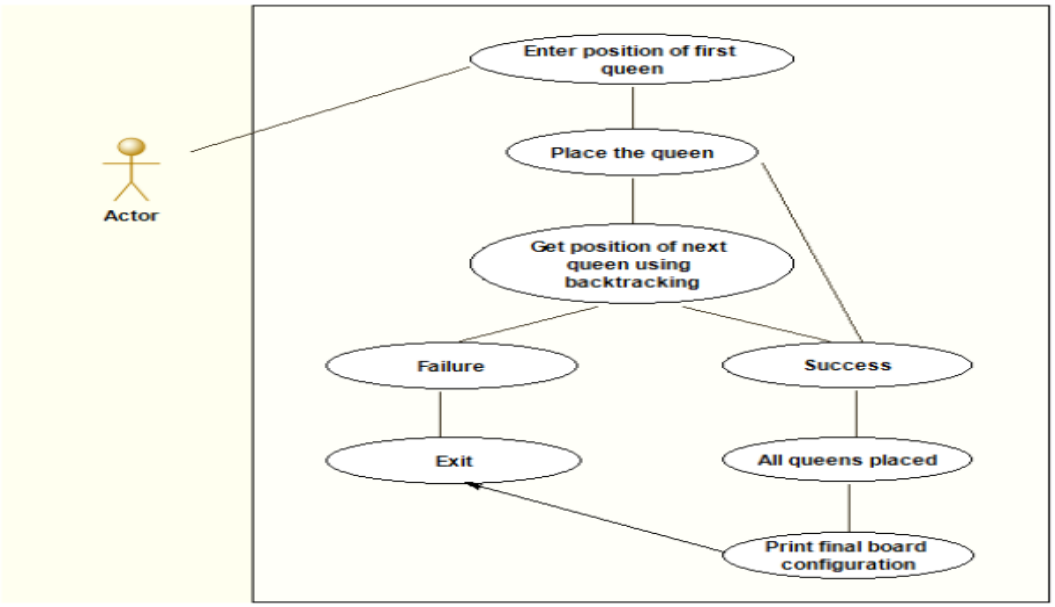

Figure 1: Usecase Diagram

State Diagram

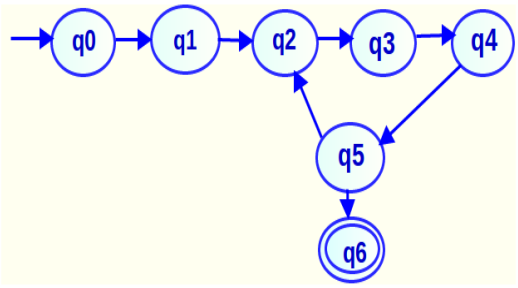

Figure 2: State Diagram

States:
q0 = Start state
q1 = Validate Input
q2 = Add queen in column
q3 = Check if the state is safe
q4 = Place queen
q5 = Check if all queens are placed
q6 = End state

Class Diagram

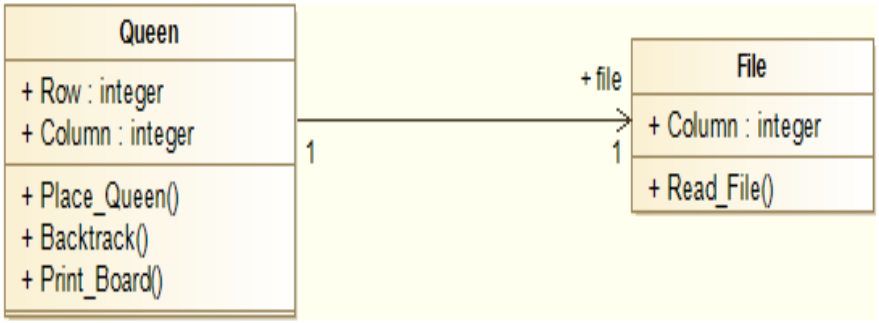

Figure 3: Class Diagram

Sequence Diagram

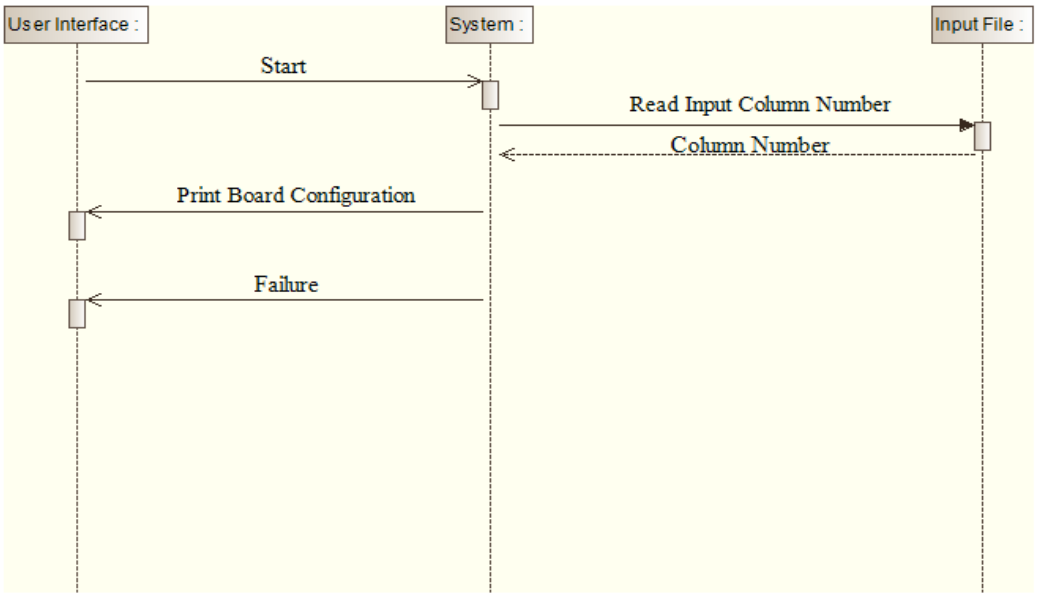

Figure 4: Sequence Diagram

ALGORITHM

1. start
2. Accept starting position of queen from XML file.
3. Place first Queen on particular position.
4. Check whether placed queen attack on another queen if yes goto step 5 if no goto step 6.
5. Solve the problem of attack.
6. place next queen on board.
7. print whole board.
8. End.

Testing:

White box Testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box

testing). White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

1. 1. First position of the queen in board.
2. 2. Size of the board.

Black box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance.

1. 1. Remaining queen's position on the board.
2. 2. Number of backtracks.

Positive Testing

Positive testing is a testing technique to show that a product or application under test does what it is supposed to do. Positive testing verifies how the application behaves for the positive set of data.

1. 1. No attack will happen on board among queen's position.
2. 2. Every queen must placed on unique position.

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	First queen is placed at appropriate position	Press Enter	First queen is placed successfully	Same as Expected
2.	Accordingly, remaining queens are placed by avoiding conflicts	Press Enter	All queens are placed at appropriate position	Same as Expected

Figure 5: Positive Testing

Negative Testing

Negative testing ensures that your application can specifically handle invalid input or unexpected user behavior.

1. 1. If two or more queen's get attacked i.e not in safe state.

Sr. No.	Test Condition	Steps to be executed	Expected Result	Actual Result
1.	Position given for placing first queen is between 0 to 7 or not	Press Enter	Yes	Yes
3.	Remaining queens are placed with conflicts	Press Enter	All queens are not placed	Same as Expected Result

Figure 6: Negative Testing

CONCLUSION:

We have implemented 8 queens problem successfully.