## HACETTEPE UNIVERSITY

## DEPARTMENT OF COMPUTER ENGINEERING BBM405



Name : Mehmet Taha

Surname : USTA

Number : 21527472

Subject : Using theorem provers to

solve constraint satisfaction problems.

Data Due: 21.06.2020 23.59

### Question-1)

### -> Import library

### -> z3 object for process

### -> add formula in Solver object

### -> return SMTLIB2 formatted benchmark for solver's assertions

### -> Return a formatted string (in Lisp-like format) with all added constraints. We say the string is in s-expression format.

### 

### s.check() -> Check whether the assertions in the given solver plus the optional assumptions are consistent or not.

### z3.sat -> means satisfiable

### s.model -> Return a model for the last ‘check()’. This function raises an exception if a model is not available (e.g., last ‘check()’ returned unsat).

### Define Boolean values (A-F)

### 

### 1. Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### 2. Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### 3. Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### 4. Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### 5. Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Question-2)

### Unrestricted Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Unrestricted Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Unrestricted Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Unrestricted Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Unrestricted Propositinal Formula

### 

### s.to\_smt2() result

### 

### s.model() result

### 

### SMT-LIB2 format

### 

### Output

### 

### Question-3)

### Define the variables

### 

|  |  |  |  |
| --- | --- | --- | --- |
| x1y1 | x1y2 | x1y3 | x1y4 |
| x2y1 | x2y2 | x2y3 | x2y4 |
| x3y1 | x3y2 | x3y3 | x3y4 |
| x4y1 | x4y2 | x4y3 | x4y4 |

### Lines declarations

### 

### 2 queens should not be on the same row

### 

### Example figure

### 

### 2 queens should not be on the same column

### 

### Example figure

### 

### 2 queens should not be on the same diagonal

### 

### Example figure

### 

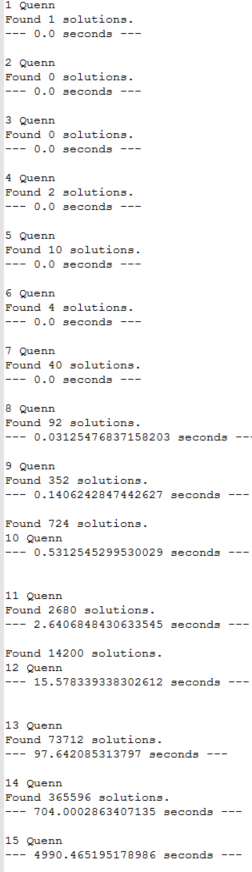
### 

### Result

### 

### 

### Question-4)



Code has been tried on python

The N- queens puzzle is the problem of placing N chess queens on an N×N 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 problem of finding all solutions to the 8-queens problem can be quite computationally expensive, as there are 4,426,165,368 (i.e., 64C8) possible arrangements of eight queens on an 8×8 board, but only 92 solutions.

As the number N increases, the complexity increases, causing the calculation to take longer

**Higher dimensions**

Find the number of non-attacking queens that can be placed in a d-dimensional chess space of size n. More than n queens can be placed in some higher dimensions