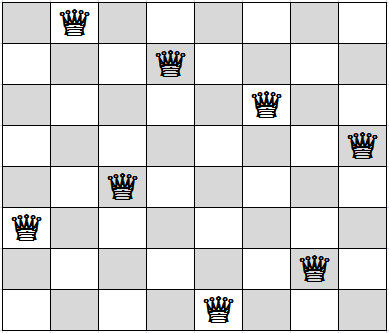
**ITCS 6150**

**Project Statement**

1. Write a program which solves N queens puzzle (8 < N < 13).
2. Input: N (number of queen)
3. Output: N×N Boolean array with 1’s showing final position of queens.
4. Program should use heuristics minimizing the search space.



**Project Members**

**Name Student ID**

Shravanti Bugade 801076033

Pratik Kontamwar 801076690

Rithesh Shetty 801058230

**INDEX**

**Content Sr No.**

**Program Structure 3**

**Global Variables In Program 4**

**Function to Compute Heuristic Function 5**

**Language and Compiler 6**

**Project Execution Steps 6 - 7**

**Program Structure**

The N-Queens Algorithm contains a logic of Random restart hill climbing search method for a given input using Python 3 compiler:

The structure of the program can be visualized as follows:

1. def generate\_board(): Function to generate board
2. def set\_variables(): Function to initialize all required variables
3. def restart\_random(): Function to run the random restart algorithm technique with sideways package
4. def next\_board(board): Function to find the next board for further processing
5. def find\_heur\_count(check\_board): Function to find the heuristics of the current board
6. def steep\_ascent\_board(board): Function to run the board on the steepest ascent algorithm
7. def output\_restart\_final(board): Function to output next stat on the screen
8. Main function: Begin program execution function. Need to enter the number of queens for which we want to apply heuristic logic

**Global Variables in Program**

1. queen\_count
2. min\_prior
3. pass\_count
4. failure\_count
5. violations
6. moves\_success
7. moves\_fail
8. moves
9. run\_count
10. counter
11. heur\_count
12. optimal\_pos

**Function to Compute Heuristic Function**

def find\_heur\_count(check\_board):

1. violations = 0
2. global queen\_count
3. for i in range(queen\_count):
   * 1. for j in range(i+1,queen\_count):
4. threat = abs(j-i)
5. diagonal\_threat = abs(check\_board[i] - check\_board[j])
6. if(check\_board[i]==check\_board[j] or threat == diagonal\_threat):

violations += 1

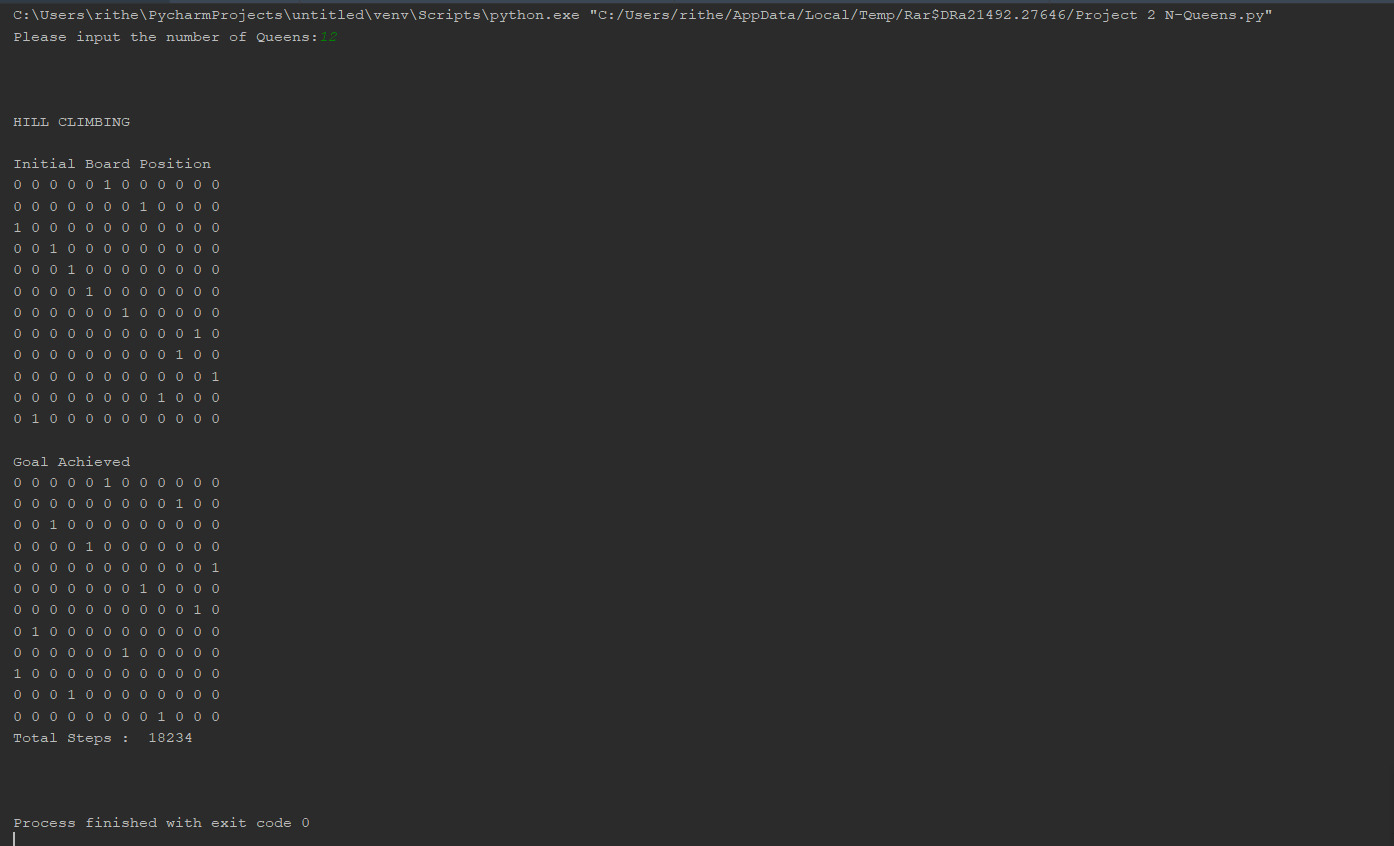
1. return violations

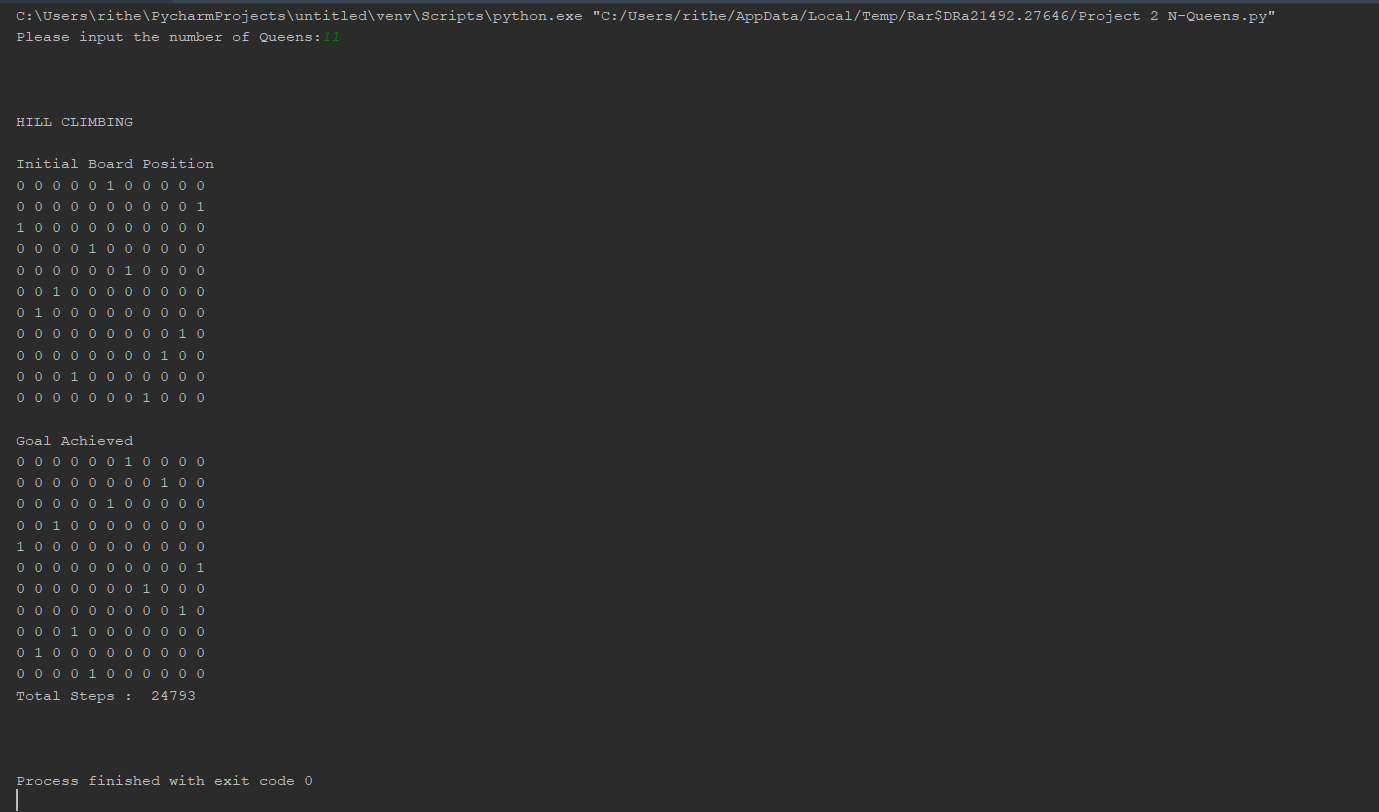
**Language and Compiler**

Python Language with use of Anaconda Compiler Tools

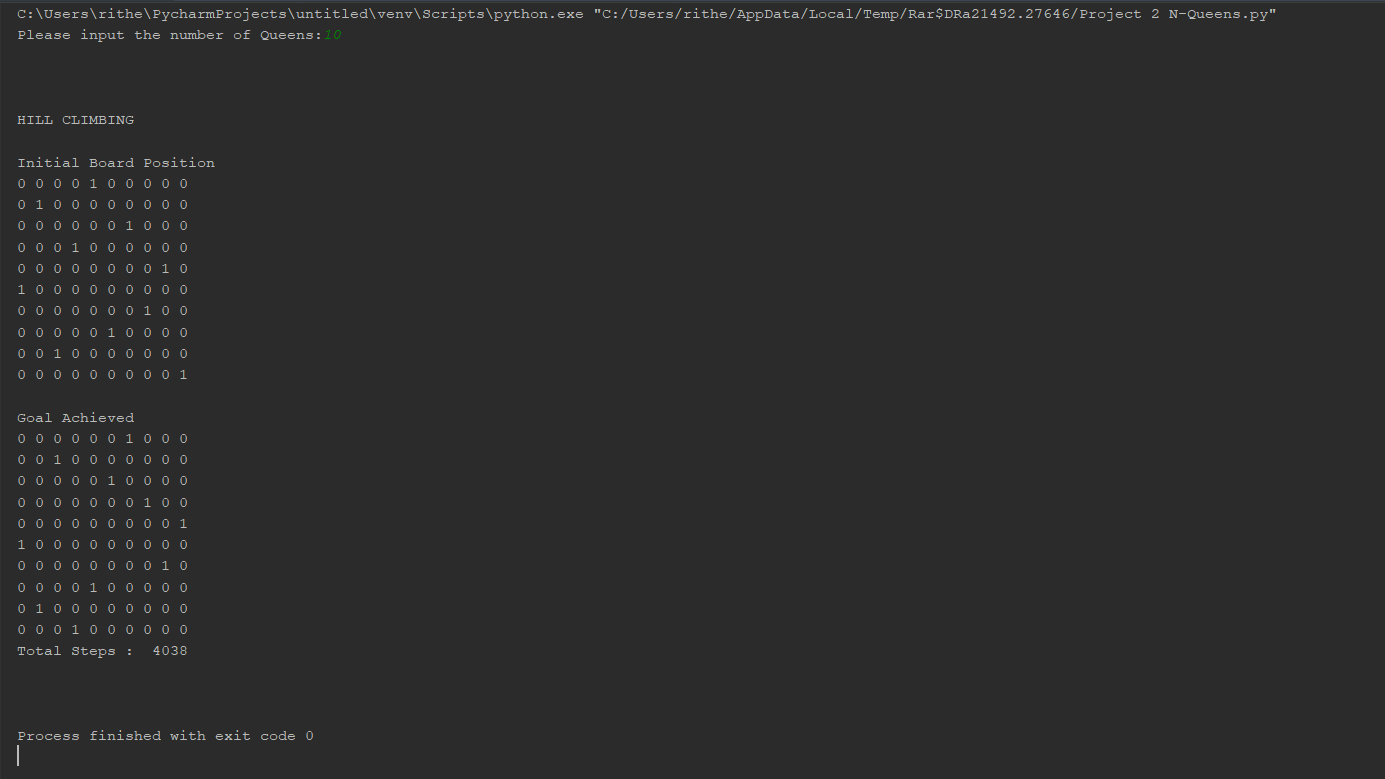
**Project Execution Steps**

Input : Enter number of queens – 12



Input : Enter number of queens – 11

Input : Enter number of queens – 10



Input : Enter number of queens – 9

