Started on	Friday, 25 April 2025, 2:32 PM
State	Finished
Completed on	Friday, 25 April 2025, 7:33 PM
Time taken	5 hours
Overdue	3 hours
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program for Bad Character Heuristic of Boyer Moore String Matching Algorithm

For example:

Input	Result
ABAAAABCD ABC	Pattern occur at shift = 5

Answer: (penalty regime: 0 %)

Reset answer

```
def preprocess_strong_suffix(shift, bpos, pat, m):
    i = m
    j = m + 1
    bpos[i] = j
    while i > 0:
        while j <= m and pat[i - 1] != pat[j - 1]:
            if shift[j] == 0:
                shift[j] = j - i
            j = bpos[j]
        i -= 1
        j -= 1
        bpos[i] = j
def preprocess_case2(shift, bpos, pat, m):
    j = bpos[0]
    for i in range(m + 1):
        if shift[i] == 0:
            shift[i] = j
        if i == j:
            j = bpos[j]
def search(text, pat):
    s = 0
    m = len(pat)
```

	Input	Expected	Got	
✔	ABAAAABCD ABC	Pattern occur at shift = 5	Pattern occur at shift = 5	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

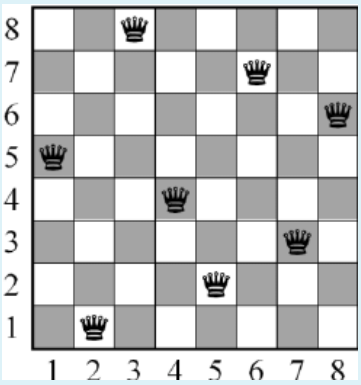
Question **2**

Correct

Mark 20.00 out of 20.00

You are given an integer **N**. For a given **N** x **N** chessboard, find a way to place **'N'** queens such that no queen can attack any other queen on the chessboard.

A queen can be attacked when it lies in the same row, column, or the same diagonal as any of the other queens. **You have to print one such configuration**.



Note :

Get the input from the user for N . The value of N must be from 1 to 8

If solution exists Print a binary matrix as output that has 1s for the cells where queens are placed

If there is no solution to the problem print "Solution does not exist"

For example:

Input	Result
5	1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0

Answer: (penalty regime: 0 %)

```
global N
N = int(input())
def printSolution(board):
    for i in range(N):
        for j in range(N):
            print(board[i][j], end = " ")
        print()
def isSafe(board, row, col):
    for i in range(col):
        if board[row][i] == 1:
            return False
    for i, j in zip(range(row, -1, -1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    for i, j in zip(range(row, N, 1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    return True
def solveNQUtil(board, col):
     if col>=N:
```

	Input	Expected	Got	
✔	5	1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0	1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0	✔
✔	2	Solution does not exist	Solution does not exist	✔
✔	8	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	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	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Create a python program to find the Hamiltonian path using Depth First Search for traversing the graph .

For example:

Test	Result
`hamiltonian.findCycle()`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`

Answer: (penalty regime: 0 %)

Reset answer

```
class Hamiltonian:
    def __init__(self, start):
        self.start = start
        self.cycle = []
        self.hasCycle = False
    def findCycle(self):
        self.cycle.append(self.start)
        self.solve(self.start)
    def solve(self, vertex):
        if vertex == self.start and len(self.cycle) == N+1:
            self.hasCycle = True
            self.displayCycle()
            return
        for i in range(len(vertices)):
            if adjacencyM[vertex][i] == 1 and visited[i] == 0:
                nbr = i
                visited[nbr] = 1
                self.cycle.append(nbr)
                self.solve(nbr)
                visited[nbr] = 0
                self.cycle.pop()
    def displayCycle(self):
```

	Test	Expected	Got	
✔	`hamiltonian.findCycle()`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to implement Boyer Moore Algorithm with Good Suffix heuristic to find pattern in given text string.

For example:

Input	Result
ABAAABAACD ABA	pattern occurs at shift = 0 pattern occurs at shift = 4

Answer: (penalty regime: 0 %)

Reset answer

```
def preprocess_strong_suffix(shift, bpos, pat, m):
    i = m
    j = m + 1
    bpos[i] = j
    while i > 0:
        while j <= m and pat[i - 1] != pat[j - 1]:
            if shift[j] == 0:
                shift[j] = j - i
            j = bpos[j]
        i -= 1
        j -= 1
        bpos[i] = j
def preprocess_case2(shift, bpos, pat, m):
    j = bpos[0]
    for i in range(m + 1):
        if shift[i] == 0:
            shift[i] = j
        if i == j:
            j = bpos[j]
def search(text, pat):
    s = 0
    m = len(pat)
```

	Input	Expected	Got	
✔	ABAAABAACD ABA	pattern occurs at shift = 0 pattern occurs at shift = 4	pattern occurs at shift = 0 pattern occurs at shift = 4	✔
✔	SaveethaEngineering Saveetha veetha	pattern occurs at shift = 2 pattern occurs at shift = 22	pattern occurs at shift = 2 pattern occurs at shift = 22	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to implement knight tour problem using warnsdorff's algorithm

For example:

Test	Input	Result
a.warnsdroff((x,y))	8 8 3 3	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]

Answer: (penalty regime: 0 %)

Reset answer

```
KNIGHT_MOVES = [(2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1)]
class KnightTour:
    def __init__(self, board_size):
        self.board_size = board_size  # tuple
        self.board = []
        for i in range(board_size[0]):
            temp = []
            for j in range(board_size[1]):
                temp.append(0)
            self.board.append(temp)  # empty cell
        self.move = 1
    def print_board(self):
        print('board:')
        for i in range(self.board_size[0]):
            print(self.board[i])
    def warnsdroff(self, start_pos, GUI=False):
        self.board[start_pos[0]][start_pos[1]] = self.move
        current_pos = start_pos
        while self.move < self.board_size[0] * self.board_size[1]:
            next_pos = self.find_next_pos(current_pos)
            if not next_pos:
```

	Test	Input	Expected	Got	
✔	a.warnsdroff((x,y))	8 8 3 3	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.