Started on	Friday, 2 May 2025, 2:14 PM
State	Finished
Completed on	Saturday, 3 May 2025, 2:12 PM
Time taken	23 hours 58 mins
Overdue	21 hours 58 mins
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python program to implement Hamiltonian circuit problem using Backtracking.

For example:

Result
Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                              for row in range(vertices)]
        self.V = vertices
    def isSafe(self, v, pos, path):
        if self.graph[ path[pos-1] ][v] == 0:
            return False
        for vertex in path:
            if vertex == v:
                return False

        return True
    def hamCycleUtil(self, path, pos):
        ######################Add your code here#################################
        if pos==self.V:
            return True
        for v in range(1,self.V):
            if self.isSafe(v,pos,path):
                path[pos]=v
                if self.hamCycleUtil(path,pos+1):
                    return True
```

	Expected	Got	
✔	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

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

```
NO_OF_CHARS = 256
def badCharHeuristic(string, size):
    ########### Add your Code Here ###############
    badChar = [-1] * NO_OF_CHARS
    for i in range(size):
        badChar[ord(string[i])] = i
    return badChar
def search(txt, pat):
    m = len(pat)
    n = len(txt)
    badChar = badCharHeuristic(pat, m)
    s = 0
    while(s <= n-m):
        j = m-1
        while j>=0 and pat[j] == txt[s+j]:
            j -= 1
        if j<0:
            print("Pattern occur at shift = {}".format(s))
            s += (m-badChar[ord(txt[s+m])] if s+m<n else 1)
        else:
            s += max(1, j-badChar[ord(txt[s+j])])
def main():
```

	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 **3**

Correct

Mark 20.00 out of 20.00

Write a python program to implement KMP (Knuth Morris Pratt).

For example:

Input	Result
ABABDABACDABABCABAB ABABCABAB	Found pattern at index 10

Answer: (penalty regime: 0 %)

Reset answer

```
def KMPSearch(pat, txt):
    ############## Add your code here #################
    lp=len(pat)
    ls=len(txt)
    lps=[0]*lp
    computeLPSArray(pat,lp,lps)
    i=0
    j=0

    while(i!=ls):
        if txt[i]==pat[j]:
            i+=1
            j+=1
        else:
            j=lps[j-1]
        if j==lp:
            print("Found pattern at index",i-j)
            j=lps[j-1]
        elif j==0:
            i+=1

def computeLPSArray(pat, M, lps):
```

	Input	Expected	Got	
✔	ABABDABACDABABCABAB ABABCABAB	Found pattern at index 10	Found pattern at index 10	✔
✔	SAVEETHAENGINEERING VEETHA	Found pattern at index 2	Found pattern at index 2	✔

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 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):
    ######################### Add your code here ############################3
        x_pos,  y_pos = start_pos
        self.board[x_pos][y_pos] = self.move

```

	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.

Question **5**

Not answered

Mark 0.00 out of 20.00

Write a python program to print the following pattern

5 4 3 2 1
5 4 3 2
5 4 3
5 4
5

For example:

Input	Result
5	5 4 3 2 1 5 4 3 2 5 4 3 5 4 5
6	6 5 4 3 2 1 6 5 4 3 2 6 5 4 3 6 5 4 6 5 6

Answer: (penalty regime: 0 %)

```
1
```