Started on	Saturday, 26 April 2025, 8:23 AM
State	Finished
Completed on	Saturday, 26 April 2025, 9:21 AM
Time taken	58 mins 40 secs
Grade	**80.00** out of 100.00

Question **1**

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):
        #############  Add your code here ###############
        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)
```

	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 **2**

Incorrect

Mark 0.00 out of 20.00

Greedy coloring doesn't always use the minimum number of colors possible to color a graph. For a graph of maximum degree `x`, greedy coloring will use at most `x+1` color. Greedy coloring can be arbitrarily bad;

Create a python program to implement graph colouring using Greedy algorithm.

For example:

Test	Result

`colorGraph(graph, n)`	`Color assigned to vertex 0 is BLUE`
	`Color assigned to vertex 1 is GREEN`
	`Color assigned to vertex 2 is BLUE`
	`Color assigned to vertex 3 is RED`
	`Color assigned to vertex 4 is RED`
	`Color assigned to vertex 5 is GREEN`

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph:
    def __init__(self, edges, n):
        self.adjList = [[] for _ in range(n)]
        # add edges to the undirected graph
        for (src, dest) in edges:
            self.adjList[src].append(dest)
            self.adjList[dest].append(src)
def colorGraph(graph, n):
    ############ Add your code here ##############
if __name__ == '__main__':
    colors = ['', 'BLUE', 'GREEN', 'RED', 'YELLOW', 'ORANGE', 'PINK',
              'BLACK', 'BROWN', 'WHITE', 'PURPLE', 'VOILET']
    edges = [(0, 1), (0, 4), (0, 5), (4, 5), (1, 4), (1, 3), (2, 3), (2, 4)]
    n = 6
    graph = Graph(edges, n)
    colorGraph(graph, n)
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 10)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a python program to implement knight tour problem

For example:

Input	Result
5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (1, 4), (0, 2)] Done!

Answer: (penalty regime: 0 %)

Reset answer

```
1  import sys
2  class KnightsTour:
3      def __init__(self, width, height):
4          self.w = width
5          self.h = height
6          self.board = []
7          self.generate_board()
8
9      def generate_board(self):
10         for i in range(self.h):
11             self.board.append([0]*self.w)
12
13     def print_board(self):
14
15         for elem in self.board:
16             print (elem)
17
18     def generate_legal_moves(self, cur_pos):
19         possible_pos = []
20         move_offsets = [(1, 2), (1, -2), (-1, 2), (-1, -2),
21                         (2, 1), (2, -1), (-2, 1), (-2, -1)]
22         x, y = cur_pos
```

	Input	Expected	Got	
✔	5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (1, 4), (0, 2)] Done!	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (1, 4), (0, 2)] Done!	✔

	Input	Expected	Got	
✔	6 6	[1, 32, 9, 18, 3, 34] [10, 19, 2, 33, 26, 17] [31, 8, 25, 16, 35, 4] [20, 11, 36, 27, 24, 15] [7, 30, 13, 22, 5, 28] [12, 21, 6, 29, 14, 23] [(0, 0), (1, 2), (0, 4), (2, 5), (4, 4), (5, 2), (4, 0), (2, 1), (0, 2), (1, 0), (3, 1), (5, 0), (4, 2), (5, 4), (3, 5), (2, 3), (1, 5), (0, 3), (1, 1), (3, 0), (5, 1), (4, 3), (5, 5), (3, 4), (2, 2), (1, 4), (3, 3), (4, 5), (5, 3), (4, 1), (2, 0), (0, 1), (1, 3), (0, 5), (2, 4), (3, 2)] Done!	[1, 32, 9, 18, 3, 34] [10, 19, 2, 33, 26, 17] [31, 8, 25, 16, 35, 4] [20, 11, 36, 27, 24, 15] [7, 30, 13, 22, 5, 28] [12, 21, 6, 29, 14, 23] [(0, 0), (1, 2), (0, 4), (2, 5), (4, 4), (5, 2), (4, 0), (2, 1), (0, 2), (1, 0), (3, 1), (5, 0), (4, 2), (5, 4), (3, 5), (2, 3), (1, 5), (0, 3), (1, 1), (3, 0), (5, 1), (4, 3), (5, 5), (3, 4), (2, 2), (1, 4), (3, 3), (4, 5), (5, 3), (4, 1), (2, 0), (0, 1), (1, 3), (0, 5), (2, 4), (3, 2)] Done!	✔

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 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 #################
    M = len(pat)
    N = len(txt)
    lps = [0]*M
    j = 0
    computeLPSArray(pat, M, lps)
    i = 0
    while (N - i) >= (M - j):
        if pat[j] == txt[i]:
            i += 1
            j += 1
        if j == M:
            print ("Found pattern at index " + str(i-j))
            j = lps[j-1]
        elif i < N and pat[j] != txt[i]:
            if j != 0:
                j = lps[j-1]
            else:
                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 **5**

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])])
```

	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.