Started on	Saturday, 10 May 2025, 2:37 PM
State	Finished
Completed on	Saturday, 10 May 2025, 3:03 PM
Time taken	25 mins 12 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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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()
        for i in range(len(vertices)):
            if adjacencyM[vertex][i]==1 and visited[i]==0:
                nbr=i
```

	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**

Correct

Mark 20.00 out of 20.00

Write a python program to implement pattern matching on the given string using Brute Force algorithm.

For example:

Test	Input	Result
BF(a1,a2)	abcaaaabbbbcccabcbabdbcsbbbbbnnnccabcba	12

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def BF(s1,s2):
############## Add your code here #############
    i=0
    j=0
    while i<len(s1) and j< len(s2):
        if s1[i]==s2[j]:
            i+=1
            j+=1
        else:
            i=i-j+1
            j=0
    if j>=len(s2):
        return i-len(s2)
    else:
        return 0

if __name__ == "__main__":
    a1=input()
```

	Test	Input	Expected	Got	
✔	BF(a1,a2)	abcaaaabbbbcccabcbabdbcsbbbbbnnnccabcba	12	12	✔

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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def KMPSearch(pat, txt):
    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:
```

	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 find minimum steps to reach to specific cell in minimum moves by knight.

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
class cell:
    def __init__(self, x = 0, y = 0, dist = 0):
        self.x = x
        self.y = y
        self.dist = dist

def isInside(x, y, N):
    if (x >= 1 and x <= N and
        y >= 1 and y <= N):
        return True
    return False
def minStepToReachTarget(knightpos,
                         targetpos, N):
    # add your code here
    dx = [2, 2, -2, -2, 1, 1, -1, -1]
    dy = [1, -1, 1, -1, 2, -2, 2, -2]
    queue = []
    queue.append(cell(knightpos[0], knightpos[1], 0))
```

	Input	Expected	Got	
✔	30	20	20	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

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

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
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.