Started on	Tuesday, 13 May 2025, 3:28 PM
State	Finished
Completed on	Tuesday, 13 May 2025, 3:44 PM
Time taken	15 mins 19 secs
Grade	**80.00** out of 100.00

Question **1**

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

```
import sys
class KnightsTour:
    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.board = []
        self.generate_board()

    def generate_board(self):
        for i in range(self.h):
            self.board.append([0]*self.w)

    def print_board(self):

        for elem in self.board:
            print (elem)

    def generate_legal_moves(self, cur_pos):
        possible_pos = []
        move_offsets = [(1, 2), (1, -2), (-1, 2), (-1, -2),
                        (2, 1), (2, -1), (-2, 1), (-2, -1)]
```

	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!	✔
✔	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 **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 #################
    M=len(pat)
    N=len(txt)
    lps=[0]*M
    computeLPSArray(pat, M, lps)
    i=0 # txt inx
    j=0 # pat indx
    while (i<N):
        if pat[j]==txt[i]:
            i+=1
            j+=1
        if j==M:
            print("Found pattern at index",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):
    len = 0
```

	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**

Incorrect

Mark 0.00 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 ###############



    def displayCycle(self):
        names = []
        for v in self.cycle:
            names.append(vertices[v])
        print(names)

```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 16)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to print the following pattern based on the given input.

Input:6
Output:

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

For example:

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

Answer: (penalty regime: 0 %)

```
n=int(input())
for i in range(n):
    for j in range(i):
        print(i,end=" ")
    if(i>=1):
        print()

```

	Input	Expected	Got	
✔	6	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5	✔
✔	5	1 2 2 3 3 3 4 4 4 4	1 2 2 3 3 3 4 4 4 4	✔

	Input	Expected	Got	
✔	8	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7 7	1 2 2 3 3 3 4 4 4 4 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7 7	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.