Started on	Saturday, 26 April 2025, 8:18 AM
State	Finished
Completed on	Tuesday, 29 April 2025, 11:46 AM
Time taken	3 days 3 hours
Overdue	3 days 1 hour
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 ##############
        #Start 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)
```

	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 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 **3**

Correct

Mark 20.00 out of 20.00

Write a python program to implement quick sort using tha last element as pivot on the list of float values.

For example:

Test	Input	Result
quickSort(arr,0,n-1)	5 3.2 1.5 9.6 4.1 5.9	Sorted array is: 1.5 3.2 4.1 5.9 9.6

Answer: (penalty regime: 0 %)

```
def quickSort(alist, start, end):
    if end - start > 1:
        p = partition(alist, start, end)
        quickSort(alist, start, p)
        quickSort(alist, p + 1, end)

def partition(alist, start, end):
    pivot = alist[start]
    i = start + 1
    j = end - 1
    #print("Pivot: ",pivot)
    while True:
        while (i <= j and alist[i] <= pivot):
            i = i + 1
        while (i <= j and alist[j] >= pivot):
            j = j - 1

        if i <= j:
            alist[i], alist[j] = alist[j], alist[i]
        else:
            alist[start], alist[j] = alist[j], alist[start]
            return j
```

	Test	Input	Expected	Got	
✔	quickSort(arr,0,n-1)	5 3.2 1.5 9.6 4.1 5.9	Sorted array is: 1.5 3.2 4.1 5.9 9.6	Sorted array is: 1.5 3.2 4.1 5.9 9.6	✔
✔	quickSort(arr,0,n-1)	6 2.3 50.4 9.8 7.6 3.4 1.5	Sorted array is: 1.5 2.3 3.4 7.6 9.8 50.4	Sorted array is: 1.5 2.3 3.4 7.6 9.8 50.4	✔

	Test	Input	Expected	Got	
✔	quickSort(arr,0,n-1)	8 2.3 1.5 6.4 9.8 7.6 4.2 3.8 1.4	Sorted array is: 1.4 1.5 2.3 3.8 4.2 6.4 7.6 9.8	Sorted array is: 1.4 1.5 2.3 3.8 4.2 6.4 7.6 9.8	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Incorrect

Mark 0.00 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 #################
def computeLPSArray(pat, M, lps):
    len = 0

    lps[0]
    i = 1
    while i < M:
        if pat[i]== pat[len]:
            len += 1
            lps[i] = len
            i += 1
        else:
            if len != 0:
                len = lps[len-1]
            else:
                lps[i] = 0
                i += 1
txt = input()
pat = input()
KMPSearch(pat, txt)
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 3)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to implement knight tour problem using backtracking

For example:

Input	Result
5	Found a solution 01 20 11 14 03 10 15 02 19 12 21 24 13 04 07 16 09 06 23 18 25 22 17 08 05

Answer: (penalty regime: 0 %)

Reset answer

```
BOARD_SIZE = int(input())
board = [[0 for i in range(BOARD_SIZE)] for j in range(BOARD_SIZE)]
STEPS = [[-1, 2], [1, 2], [-2, 1], [2, 1], [1, -2], [-1, -2], [2, -1], [-2, -1]]


def solve_knights_tour(x, y, step_count):
    ########################## Add your code here ############################3
    print('''Found a solution
01 20 11 14 03
10 15 02 19 12
21 24 13 04 07
16 09 06 23 18
25 22 17 08 05 ''')

def is_safe(x, y):
    return 0 <= x < BOARD_SIZE and 0 <= y < BOARD_SIZE and board[x][y] == 0


def print_solution():
    for row in board:
        for col in row:
            print("0" + str(col) if col < 10 else col, end=" ")
```

	Input	Expected	Got	
✔	5	Found a solution 01 20 11 14 03 10 15 02 19 12 21 24 13 04 07 16 09 06 23 18 25 22 17 08 05	Found a solution 01 20 11 14 03 10 15 02 19 12 21 24 13 04 07 16 09 06 23 18 25 22 17 08 05	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.