Started on	Friday, 16 May 2025, 3:43 PM
State	Finished
Completed on	Friday, 16 May 2025, 4:14 PM
Time taken	31 mins 12 secs
Grade	**80.00** out of 100.00

Question **1**

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):

    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 **2**

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):
        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)
                visited[nbr] = 0
```

	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 **3**

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

```python
NO_OF_CHARS = 256
def badCharHeuristic(string, size):

    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.

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

```python
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):

    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))
    visited = [[False for i in range(N + 1)] for j in range(N + 1)]
    visited[knightpos[0]][knightpos[1]] = True
    while(len(queue) > 0):
```

	Input	Expected	Got	
✓	30	20	20	✓

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 implement quick sort using random pivot value.

For example:

Input	Result
6 10 7 8 9 1 5	[1, 5, 7, 8, 9, 10]

Answer: (penalty regime: 0 %)