Started on	Wednesday, 12 March 2025, 3:07 PM
State	Finished
Completed on	Wednesday, 12 March 2025, 3:42 PM
Time taken	34 mins 56 secs
Grade	**100.00** out of 100.00

Question **1**

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):
    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():
    txt = input()                       #"ABAAABCD"
```

	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 **2**

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):
    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 **3**

Correct

Mark 20.00 out of 20.00

Write a python program to check whether Hamiltonian path exits in the given graph.

For example:

Test	Result
`Hamiltonian_path(adj, N)`	YES

Answer: (penalty regime: 0 %)

Reset answer

```
def Hamiltonian_path(adj, N):
    dp = [[False for i in range(1 << N)] for j in range(N)]
    for i in range(N):
        dp[i][1 << i] = True
    for i in range(1 << N):
        for j in range(N):
            if ((i & (1 << j)) != 0):
                for k in range(N):
                    if ((i & (1 << k)) != 0 and adj[k][j] == 1 and j != k and dp[k][i ^ (1 << j)]):
                        dp[j][i] = True
                        break
    for i in range(N):
        if (dp[i][(1 << N) - 1]):
            return N

adj = [ [ 0, 1, 1, 1, 0 ] ,
        [ 1, 0, 1, 0, 1 ],
        [ 1, 1, 0, 1, 1 ],
        [ 1, 0, 1, 0, 0 ] ]

N = len(adj)
```

	Test	Expected	Got	
✔	`Hamiltonian_path(adj, N)`	YES	YES	✔

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 sort unsorted numbers using Multi-key quicksort

For example:

Test	Input	Result
quick_sort_3partition(nums, 0, len(nums)-1)	5 4 3 5 1 2	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]
quick_sort_3partition(nums, 0, len(nums)-1)	6 21 10 3 65 4 8	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]

Answer: (penalty regime: 0 %)

```
def partition(nums,l,r):
    pivot = nums[r]
    ptr = l - 1
    for i in range(l, r):
        if nums[i] <= pivot:
            ptr += 1
            nums[ptr], nums[i] = nums[i], nums[ptr]
    nums[ptr + 1], nums[r] = nums[r], nums[ptr + 1]
    return ptr + 1
def quick_sort_3partition(nums,l,r):
    if l < r:
        pi = partition( nums,l, r)
        quick_sort_3partition( nums,l, pi - 1)
        quick_sort_3partition(nums,pi + 1, r)
    return nums

n = int(input())
nums = []

for _ in range(n):
    num = int(input())
    nums.append(num)
```

	Test	Input	Expected	Got	
✔	quick_sort_3partition(nums, 0, len(nums)-1)	5 4 3 5 1 2	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]	✔
✔	quick_sort_3partition(nums, 0, len(nums)-1)	6 21 10 3 65 4 8	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]	Original list: [21, 10, 3, 65, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [3, 4, 8, 10, 21, 65]	✔

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to implement knight tour problem using warnsdorff's algorithm

For example:

Test	Input	Result
a.warnsdroff((x,y))	8 8 3 3	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]

Answer: (penalty regime: 0 %)

Reset answer

```
KNIGHT_MOVES = [(2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1)]
class KnightTour:
    def __init__(self, board_size):
        self.board_size = board_size  # tuple
        self.board = []
        for i in range(board_size[0]):
            temp = []
            for j in range(board_size[1]):
                temp.append(0)
            self.board.append(temp) # empty cell
        self.move = 1

    def print_board(self):
        print('board:')
        for i in range(self.board_size[0]):
            print(self.board[i])

    def warnsdroff(self, start_pos, GUI=False):
        x_pos,y_pos=start_pos
        self.board[x_pos][y_pos]=self.move
        if not GUI:
```

	Test	Input	Expected	Got	
✔	a.warnsdroff((x,y))	8 8 3 3	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.