Started on	Wednesday, 14 May 2025, 9:17 AM
State	Finished
Completed on	Wednesday, 14 May 2025, 12:13 PM
Time taken	2 hours 56 mins
Overdue	56 mins 56 secs
Grade	**80.00** out of 100.00

Question **1**

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

```
def BF(s1,s2):
    i=len(s1)
    j=len(s2)
    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 j+5
    else:
        return 0


if __name__ == "__main__":
    a1=input()
    a2=input()
    b=BF(a1,a2)
    print(b)
```

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

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 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 True
    return False

```

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

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):

    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 **4**

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:
            while self.move<=self.board_size[0]*self.board_size[1]:
```

	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.

Question **5**

Not answered

Mark 0.00 out of 20.00

Write a python program using quick sort to sort the second half of the given list of values.

For example:

Input	Result
7 5 1 3 9 8 2 7	[2, 1, 3, 5, 7, 8, 9]
7 2 6 30 14 8 9 7	[2, 6, 7, 14, 8, 9, 30]

Answer: (penalty regime: 0 %)

```
1
```