Started on	Wednesday, 29 January 2025, 3:19 PM
State	Finished
Completed on	Wednesday, 29 January 2025, 3:58 PM
Time taken	39 mins 30 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)	abcaaaabbbbcccabcbabdbcsbbbbbnnn ccabcba	12

Answer: (penalty regime: 0 %)

Reset answer

```
def BF(s1,s2):
    i=0;
    j=0;
    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 i-len(s2)
    else:
        return 0;
##############  Add your code here #############
if __name__ == "__main__":
    a1=input()
    a2=input()
    b=BF(a1,a2)
    print(b)

```

	Test	Input	Expected	Got	
✓	BF(a1,a2)	abcaaaabbbbcccabcbabdbcsbbbbbnnn ccabcba	12	12	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Not answered

Mark 0.00 out of 20.00

Write a python program to implement quick sort using last element as pivot on the given list of integers.

For example:

Test	Input	Result
`quickSort(arr,0,n-1)`	6 21 54 30 12 10 6	Sorted array is: 6 10 12 21 30 54

Answer: (penalty regime: 0 %)

Question **3**

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 **4**

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
    ########### Add your Code Here ###############
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 **5**

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

```
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.