Started on	Monday, 30 June 2025, 2:01 PM
State	Finished
Completed on	Monday, 30 June 2025, 4:39 PM
Time taken	2 hours 38 mins
Overdue	38 mins 24 secs
Grade	**100.00** 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):
    m=len(s1)
    n=len(s2)
    for i in range(m-n+1):
        j=0
        while j<n and s1[i+j]==s2[j]:
            j+=1
        if j==n:
            return i
    return -1
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! ✔

Marks for this submission: 20.00/20.00.

Question **2**

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):
    ########### Add your Code Here ###############
    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():
```

	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 **3**

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):
    ########################## Add your code here ############################3
    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)]
```

	Input	Expected	Got	
✔	30	20	20	✔

Passed all tests! ✔

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program to implement the quick sort using recursion on the given list of float values.

For example:

Input	Result
5 6.3 1.2 4.6 5.8 9.7	pivot: 9.7 pivot: 5.8 pivot: 4.6 [1.2, 4.6, 5.8, 6.3, 9.7]
6 2.3 7.8 9.5 4.2 3.6 5.4	pivot: 5.4 pivot: 3.6 pivot: 7.8 [2.3, 3.6, 4.2, 5.4, 7.8, 9.5]

Answer: (penalty regime: 0 %)

```
def partition(l, r, nums):
    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 quicksort(l, r, nums):
    if l < r:
        pi = partition(l, r, nums)
        print(f"pivot:  {nums[pi]}")
        quicksort(l, pi - 1, nums)
        quicksort(pi + 1, r, nums)
    return nums

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

for _ in range(n):
    num = float(input())
```

	Input	Expected	Got	
✔	5 6.3 1.2 4.6 5.8 9.7	pivot: 9.7 pivot: 5.8 pivot: 4.6 [1.2, 4.6, 5.8, 6.3, 9.7]	pivot: 9.7 pivot: 5.8 pivot: 4.6 [1.2, 4.6, 5.8, 6.3, 9.7]	✔
✔	6 2.3 7.8 9.5 4.2 3.6 5.4	pivot: 5.4 pivot: 3.6 pivot: 7.8 [2.3, 3.6, 4.2, 5.4, 7.8, 9.5]	pivot: 5.4 pivot: 3.6 pivot: 7.8 [2.3, 3.6, 4.2, 5.4, 7.8, 9.5]	✔
✔	4 3.2 6.4 8.7 1.5	pivot: 1.5 pivot: 3.2 pivot: 6.4 [1.5, 3.2, 6.4, 8.7]	pivot: 1.5 pivot: 3.2 pivot: 6.4 [1.5, 3.2, 6.4, 8.7]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Create a python program to implement Hamiltonian circuit problem using Backtracking.

For example:

Result
Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                      for row in range(vertices)]
        self.V = vertices
    def isSafe(self, v, pos, path):
        if self.graph[ path[pos-1] ][v] == 0:
            return False
        for vertex in path:
            if vertex == v:
                return False

        return True
    def hamCycleUtil(self, path, pos):
        ######################Add your code here###################################
        if pos==self.V:
            return True
        for v in range(1,self.V):
            if self.isSafe(v,pos,path):
                path[pos]=v
                if self.hamCycleUtil(path,pos+1):
                    return True
```

	Expected	Got	
✔	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.