Started on	Thursday, 24 April 2025, 2:49 PM
State	Finished
Completed on	Thursday, 24 April 2025, 6:07 PM
Time taken	3 hours 18 mins
Overdue	1 hour 18 mins
Grade	**80.00** out of 100.00

Question **1**

Incorrect

Mark 0.00 out of 20.00

Write a Python program to sort unsorted numbers using Random Pivot Quick Sort. Picks the random index as the pivot

For example:

Test	Input	Result
quick_sort_random(nums, 0, len(nums))	5 1 2 65 4 9	Original list: [1, 2, 65, 4, 9] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 4, 9, 65]
quick_sort_random(nums, 0, len(nums))	6 32 10 5 6 4 8	Original list: [32, 10, 5, 6, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [4, 5, 6, 8, 10, 32]

Answer: (penalty regime: 0 %)

```
import random
def partition(list1,start,end):
    pivot_index=random.randint(start,end)
    pivot=list1[pivot_index]
    list1[pivot_index],list1[start]=list1[start],list1[pivot_index]
    i=start+1
    j=end
    while True:
        while (i<=j and list1[i]<=pivot):
            i=i+1
        while (i<=j and list1[j]>=pivot):
            j=j-1
        if i<=j:
            list1[i],list1[j]=list1[j],list1[i]
        else:
            list1[start],list1[j]=list1[j],list1[start]
            return j

def quickSort(list1,start,end):
    if start<end:
        pi=partition(list1,start,end)
        quickSort(list1,start,pi)
```

	Test	Input	Expected	Got	
✖	quick_sort_random(nums, 0, len(nums))	5 1 2 65 4 9	Original list: [1, 2, 65, 4, 9] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 4, 9, 65]	Original list: [1, 2, 65, 4, 9] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 4, 9, 65] ***Run error*** Traceback (most recent call last): File "__tester__.python3", line 73, in <module> quick_sort_random(nums, 0, len(nums)) NameError: name 'quick_sort_random' is not defined	✖

Testing was aborted due to error.
Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/20.00.

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

```
#Correct the Errors and add the required logic to get the output.
def KMPSearch(pat, txt):
#start here
    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

```

	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 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

```
#Correct the Errors and the required logic in the following code to get the output.

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

	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

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###################################
        #Start here
        if pos == self.V:
            if self.graph[ path[pos-1] ][ path[0] ] == 1:
                return True
            else:
                return False
        for v in range(1,self.V):
```

	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.

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):
    # add your code here
    #Start here
    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
```

	Input	Expected	Got	
✔	30	20	20	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.