| | |
|---|---|
| **Started on** | Tuesday, 13 May 2025, 3:18 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 13 May 2025, 3:43 PM |
| **Time taken** | 24 mins 59 secs |
| **Grade** | **80.00** out of 100.00 |

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to implement  Boyer Moore Algorithm with Good Suffix heuristic to find pattern in given text string.

**For example:**

| Input | Result |
|---|---|
| ABAAABAACD<br>ABA | pattern occurs at shift = 0<br>pattern occurs at shift = 4 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
def preprocess_strong_suffix(shift, bpos, pat, m):

    i = m
    j = m + 1
    bpos[i] = j
    while i > 0:
        while j <= m and pat[i - 1] != pat[j - 1]:
            if shift[j] == 0:
                shift[j] = j - i
            j = bpos[j]
        i -= 1
        j -= 1
        bpos[i] = j

def preprocess_case2(shift, bpos, pat, m):
    j = bpos[0]
    for i in range(m + 1):
        if shift[i] == 0:
            shift[i] = j
        if i == j:
            j = bpos[j]
def search(text, pat):
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | ABAAABAACD<br>ABA | pattern occurs at shift = 0<br>pattern occurs at shift = 4 | pattern occurs at shift = 0<br>pattern occurs at shift = 4 | ✓ |
| ✓ | SaveethaEngineering Saveetha<br>veetha | pattern occurs at shift = 2<br>pattern occurs at shift = 22 | pattern occurs at shift = 2<br>pattern occurs at shift = 22 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

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<br>0 1 2 4 3 0 |

**Answer:** (penalty regime: 0 %)

Reset answer

```python
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):

        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):
            if self.isSafe(v, pos, path) == True:
```

| | **Expected** | **Got** | |
|---|---|---|---|
| ✓ | Solution Exists: Following is one Hamiltonian Cycle<br>0 1 2 4 3 0 | Solution Exists: Following is one Hamiltonian Cycle<br>0 1 2 4 3 0 | ✓ |

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 pattern matching on the given string using Brute Force algorithm.

**For example:**

| Test | Input | Result |
|---|---|---|
| BF(a1,a2) | abcaaaabbbbcccabcbabdbcsbbbbbnnn<br>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

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

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✓ | BF(a1,a2) | abcaaaabbbbcccabcbabdbcsbbbbbnnn<br>ccabcba | 12 | 12 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Not answered

Mark 0.00 out of 20.00

Write a short recursive Python function that finds the minimum and maximum values in a sequence without using any loops.

**For example:**

| Input | Result |
|---|---|
| 4<br>51<br>20<br>31<br>47 | 51 |
| 4<br>12<br>20<br>5<br>6 | 20 |

**Answer:** (penalty regime: 0 %)

```

```

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to implement knight tour problem using backtracking

**For example:**

| Input | Result |
|---|---|
| 5 | Found a solution<br>01 20 11 14 03<br>10 15 02 19 12<br>21 24 13 04 07<br>16 09 06 23 18<br>25 22 17 08 05 |

**Answer:** (penalty regime: 0 %)

Reset answer

```python
BOARD_SIZE = int(input())
board = [[0 for i in range(BOARD_SIZE)] for j in range(BOARD_SIZE)]
STEPS = [[-1, 2], [1, 2], [-2, 1], [2, 1], [1, -2], [-1, -2], [2, -1], [-2, -1]]


def solve_knights_tour(x, y, step_count):

    if step_count > BOARD_SIZE * BOARD_SIZE:
        return True
    for step in STEPS:
        next_x = x + step[0]
        next_y = y + step[1]
        if is_safe(next_x, next_y):
            board[next_x][next_y] = step_count
            if solve_knights_tour(next_x, next_y, step_count + 1):
                return True
            board[next_x][next_y] = 0
    return False

def is_safe(x, y):
    return 0 <= x < BOARD_SIZE and 0 <= y < BOARD_SIZE and board[x][y] == 0

```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✓ | 5 | Found a solution<br>01 20 11 14 03<br>10 15 02 19 12<br>21 24 13 04 07<br>16 09 06 23 18<br>25 22 17 08 05 | Found a solution<br>01 20 11 14 03<br>10 15 02 19 12<br>21 24 13 04 07<br>16 09 06 23 18<br>25 22 17 08 05 | ✓ |

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.