CHAITRA SAMANT

231070055

DAA LAB 6A

1. **AIM:**
   Consider grades received by 20 students, like AA, AB, BB, ..., FF of each student. Computer the Longest common sequence of grades among students.

2. **PROGRAM:**

Example of Generated Dataset

```
sequences = [
    ["AA", "AB", "BB", "CC", "DD", "EE", "FF", "AA", "BB", "CC"],
    ["BB", "CC", "DD", "EE", "FF", "AA", "AB", "AA", "BB", "CC"],
    ["AA", "CC", "DD", "EE", "FF", "AA", "AB", "BB", "BB", "CC"],
    ["AA", "AB", "CC", "DD", "EE", "AA", "BB", "FF", "DD", "EE"],
    ["BB", "CC", "DD", "AA", "AB", "EE", "FF", "CC", "AA", "BB"],
    ["AA", "AB", "BB", "DD", "FF", "CC", "DD", "AA", "AB", "CC"],
    ["CC", "DD", "EE", "AA", "AB", "BB", "BB", "CC", "DD", "EE"],
    ["AA", "BB", "CC", "DD", "EE", "FF", "AA", "AB", "CC", "DD"],
    ["BB", "CC", "AA", "AB", "FF", "EE", "DD", "CC", "AA", "AB"],
    ["CC", "DD", "EE", "AA", "AB", "BB", "AA", "CC", "DD", "FF"],
    ["AA", "AB", "BB", "CC", "DD", "EE", "FF", "AA", "BB", "CC"],
    ["AB", "BB", "CC", "DD", "EE", "AA", "CC", "DD", "FF", "AA"],
    ["AA", "CC", "DD", "FF", "EE", "BB", "CC", "AA", "AB", "EE"],
    ["BB", "CC", "DD", "AA", "EE", "AB", "CC", "FF", "AA", "AB"],
    ["AA", "AB", "BB", "CC", "FF", "CC", "DD", "AA", "EE", "BB"],
    ["CC", "DD", "AA", "AB", "EE", "AA", "BB", "CC", "DD", "FF"],
    ["AA", "BB", "CC", "DD", "EE", "FF", "AA", "BB", "CC", "DD"],
    ["BB", "CC", "DD", "AA", "AB", "CC", "DD", "EE", "FF", "AA"],
    ["AA", "CC", "DD", "EE", "FF", "BB", "CC", "AA", "AB", "BB"],
    ["AB", "BB", "CC", "DD", "EE", "FF", "AA", "AB", "BB", "CC"]
]
```

```python
class LCS:
    def compute(self, P, Q):
        n,m = len(P),len(Q)

        arr = []
        for _ in range(n + 1):
            row = [0] * (m + 1)
            arr.append(row)

        def LCS_recursive(n, m):

            if arr[n][m] != 0:
                return arr[n][m]

            if n == 0 or m == 0:
                result = 0
            elif P[n-1] == Q[m-1]:
                result = 1 + LCS_recursive(n-1, m-1)
            else:
                tmp1 = LCS_recursive(n-1, m)
                tmp2 = LCS_recursive(n, m-1)
                result = max(tmp1, tmp2)


            arr[n][m] = result
            return result


        result = LCS_recursive(n, m)

        i, j = n, m
        lcs_sequence = []
        while i > 0 and j > 0:
            if P[i - 1] == Q[j - 1]:
                lcs_sequence.append(P[i - 1])
                i -= 1
                j -= 1
            elif arr[i - 1][j] >= arr[i][j - 1]:
                i -= 1
            else:
                j -= 1
        return lcs_sequence[::-1]

import re

class LCSFinder:
    def __init__(self, lcs_algorithm):
        self.lcs_algorithm = lcs_algorithm
    def validate_sequences(self, sequences):
        for seq in sequences:
            if not seq:
                print("Error: Sequence is empty.")
                exit()
            for element in seq:
                if not isinstance(element, str):
                    print("Error: All elements in the sequences must be
strings.")
                    exit()


                if not re.match("^[a-zA-Z0-9]*$", element):
                    print("Error: Special characters are not allowed.")
                    exit()

        is_lowercase = all(element.islower() for seq in sequences
                       for element in seq)
        is_uppercase = all(element.isupper() for seq in sequences

      for element in seq)

        if not (is_lowercase or is_uppercase):
            print("Error: Inconsistent case across sequences.")
            exit()

    def find(self, sequences):
        self.validate_sequences(sequences)

        common_subsequence = sequences[0]
        for seq in sequences[1:]:
            common_subsequence =
self.lcs_algorithm.compute(common_subsequence, seq)
            if not common_subsequence:
                break
        return common_subsequence
```

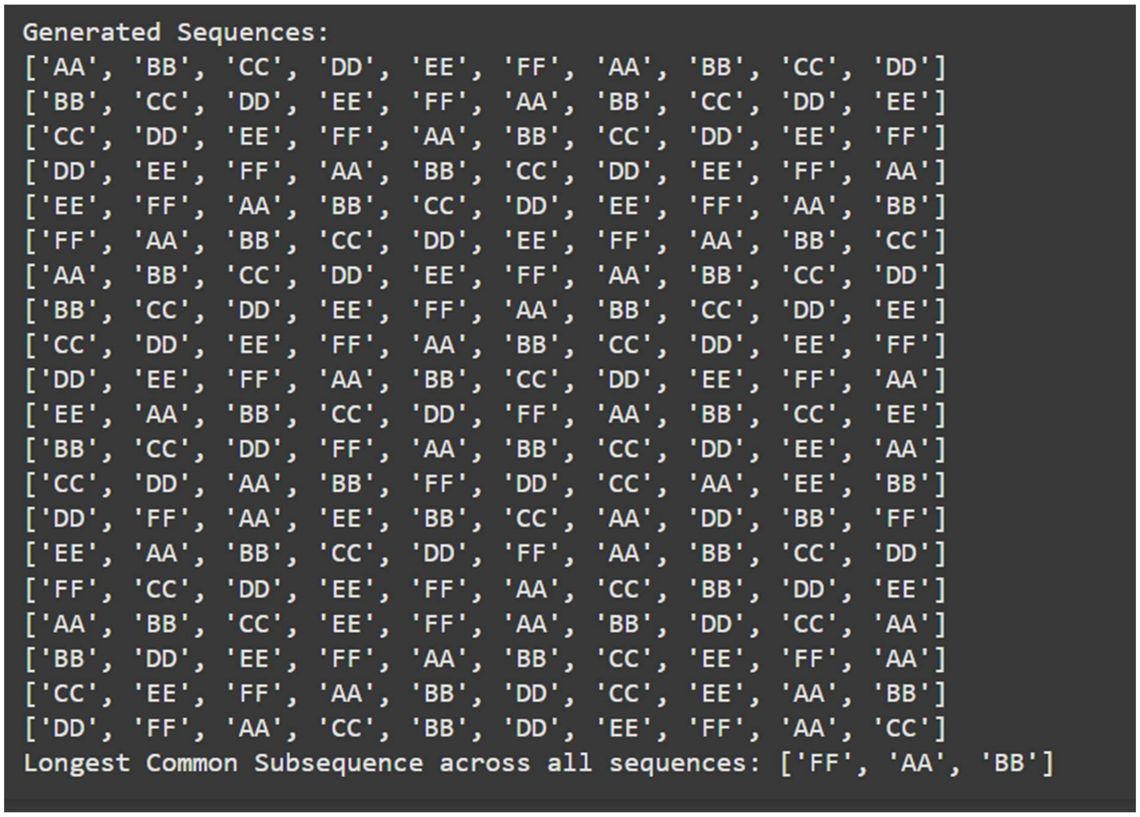
```python
lcs_algorithm = LCS()
lcs_finder = LCSFinder(lcs_algorithm)
longest_common_subsequence = lcs_finder.find(sequences)

print("Generated Sequences:")
for seq in sequences:
    print(seq)
print("Longest Common Subsequence across all sequences:",
longest_common_subsequence)
```

3. TESTCASES

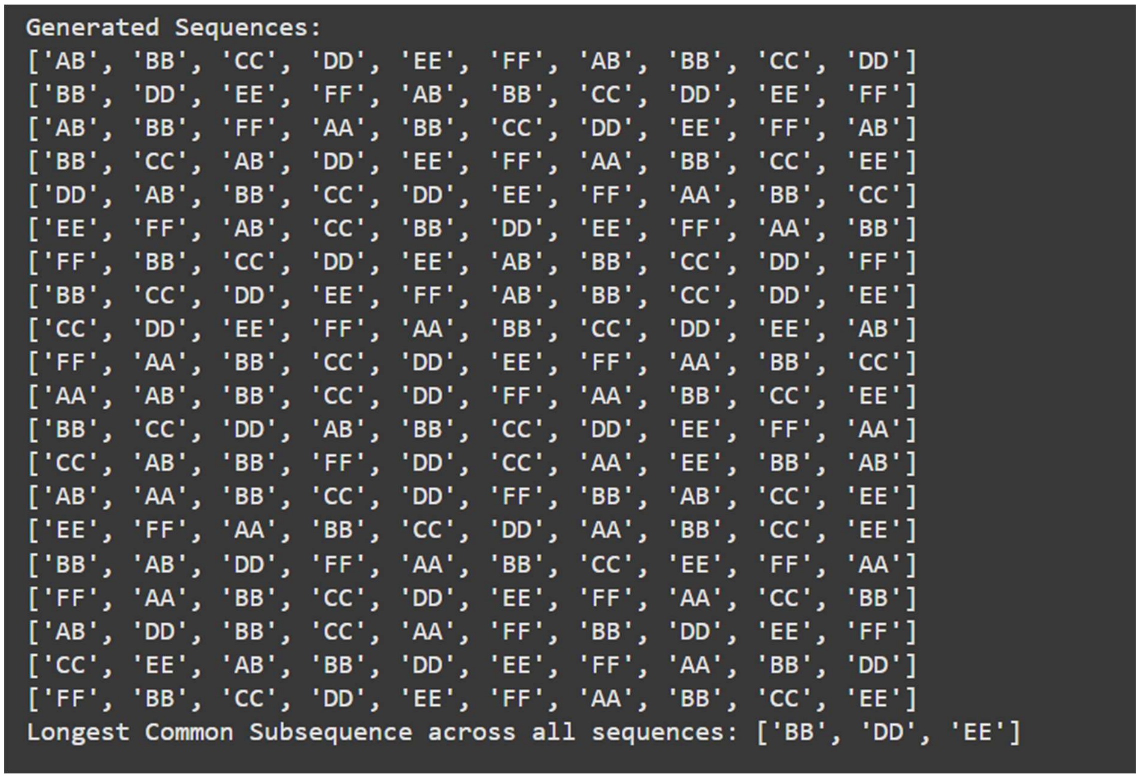
POSITIVE

- Valid TC 1



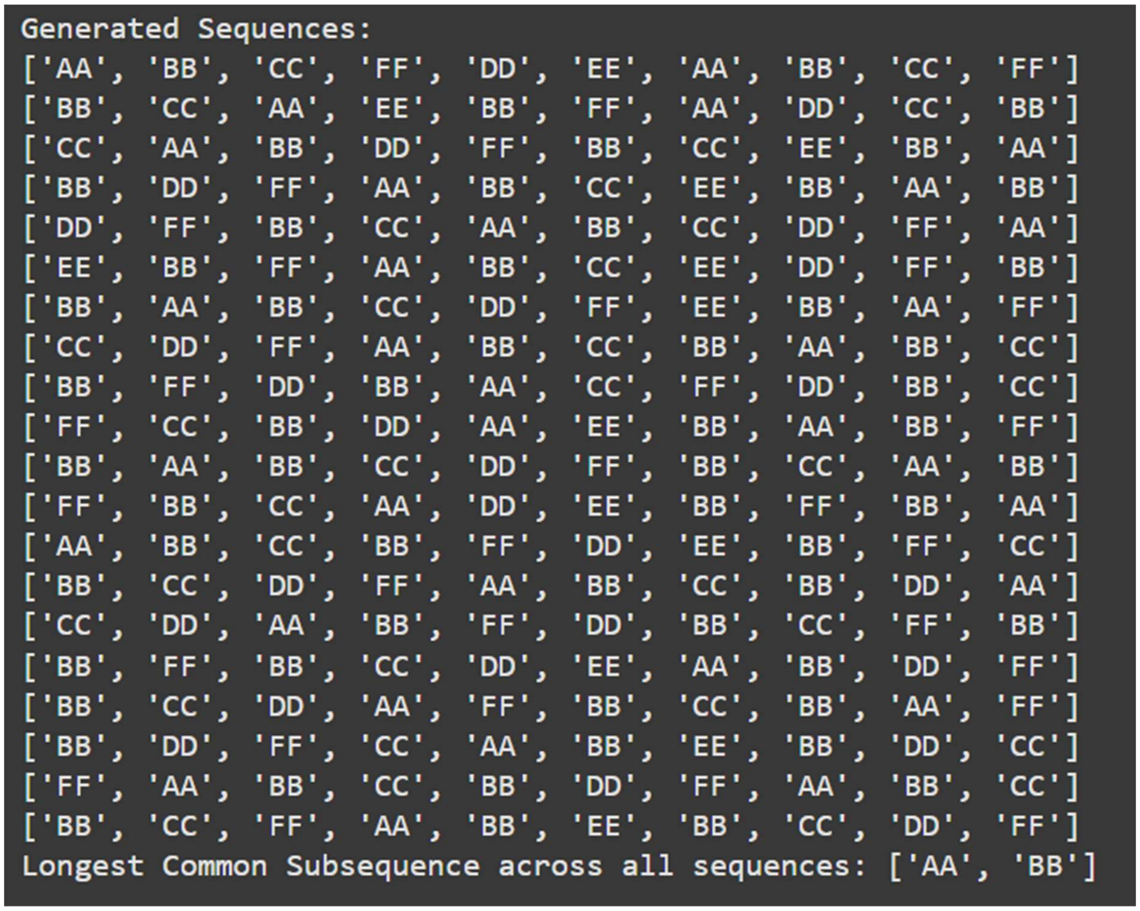
```
Generated Sequences:
['AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD']
['BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE']
['CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF']
['DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA']
['EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB']
['FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC']
['AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD']
['BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE']
['CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF']
['DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA']
['EE', 'AA', 'BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'EE']
['BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'AA']
['CC', 'DD', 'AA', 'BB', 'FF', 'DD', 'CC', 'AA', 'EE', 'BB']
['DD', 'FF', 'AA', 'EE', 'BB', 'CC', 'AA', 'DD', 'BB', 'FF']
['EE', 'AA', 'BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'DD']
['FF', 'CC', 'DD', 'EE', 'FF', 'AA', 'CC', 'BB', 'DD', 'EE']
['AA', 'BB', 'CC', 'EE', 'FF', 'AA', 'BB', 'DD', 'CC', 'AA']
['BB', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'EE', 'FF', 'AA']
['CC', 'EE', 'FF', 'AA', 'BB', 'DD', 'CC', 'EE', 'AA', 'BB']
['DD', 'FF', 'AA', 'CC', 'BB', 'DD', 'EE', 'FF', 'AA', 'CC']
Longest Common Subsequence across all sequences: ['FF', 'AA', 'BB']
```

- Valid TC2



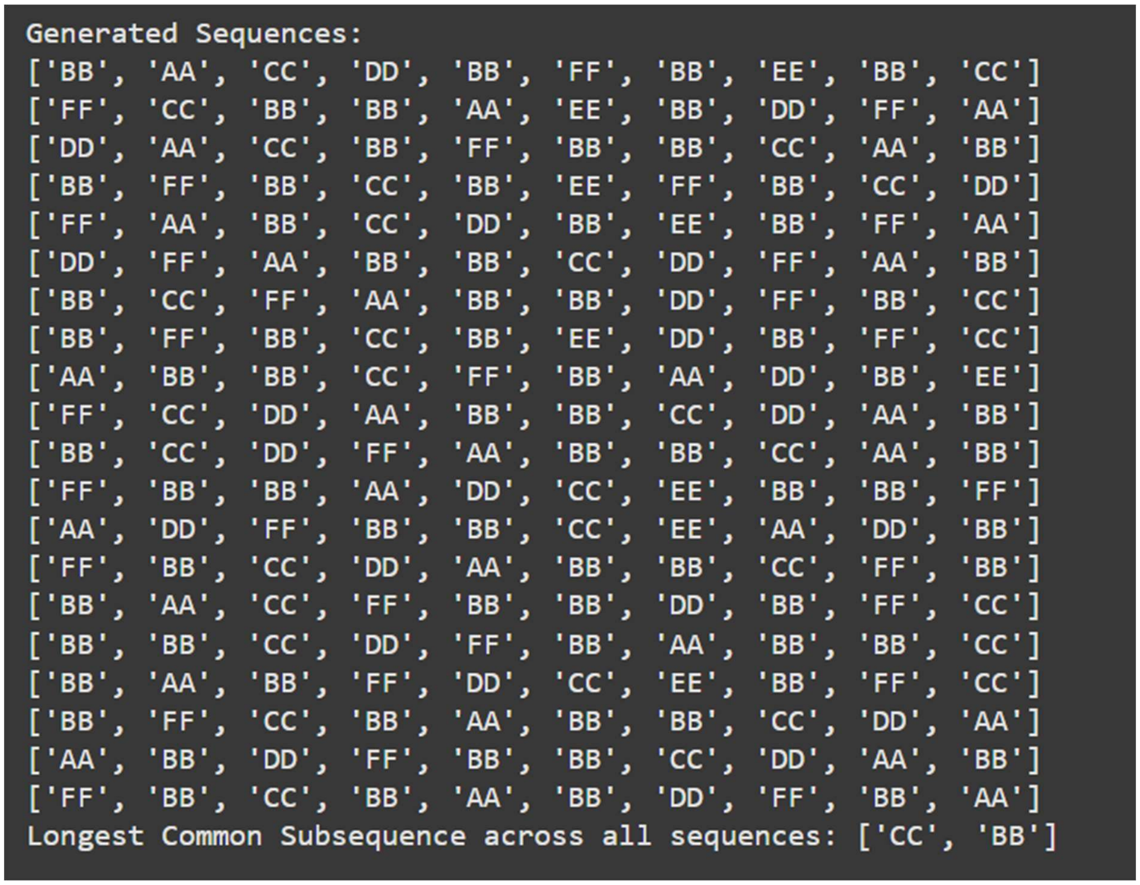
```
Generated Sequences:
['AB', 'BB', 'CC', 'DD', 'EE', 'FF', 'AB', 'BB', 'CC', 'DD']
['BB', 'DD', 'EE', 'FF', 'AB', 'BB', 'CC', 'DD', 'EE', 'FF']
['AB', 'BB', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AB']
['BB', 'CC', 'AB', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'EE']
['DD', 'AB', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC']
['EE', 'FF', 'AB', 'CC', 'BB', 'DD', 'EE', 'FF', 'AA', 'BB']
['FF', 'BB', 'CC', 'DD', 'EE', 'AB', 'BB', 'CC', 'DD', 'FF']
['BB', 'CC', 'DD', 'EE', 'FF', 'AB', 'BB', 'CC', 'DD', 'EE']
['CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'AB']
['FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC']
['AA', 'AB', 'BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'EE']
['BB', 'CC', 'DD', 'AB', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA']
['CC', 'AB', 'BB', 'FF', 'DD', 'CC', 'AA', 'EE', 'BB', 'AB']
['AB', 'AA', 'BB', 'CC', 'DD', 'FF', 'BB', 'AB', 'CC', 'EE']
['EE', 'FF', 'AA', 'BB', 'CC', 'DD', 'AA', 'BB', 'CC', 'EE']
['BB', 'AB', 'DD', 'FF', 'AA', 'BB', 'CC', 'EE', 'FF', 'AA']
['FF', 'AA', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'CC', 'BB']
['AB', 'DD', 'BB', 'CC', 'AA', 'FF', 'BB', 'DD', 'EE', 'FF']
['CC', 'EE', 'AB', 'BB', 'DD', 'EE', 'FF', 'AA', 'BB', 'DD']
['FF', 'BB', 'CC', 'DD', 'EE', 'FF', 'AA', 'BB', 'CC', 'EE']
Longest Common Subsequence across all sequences: ['BB', 'DD', 'EE']
```

- Valid TC3



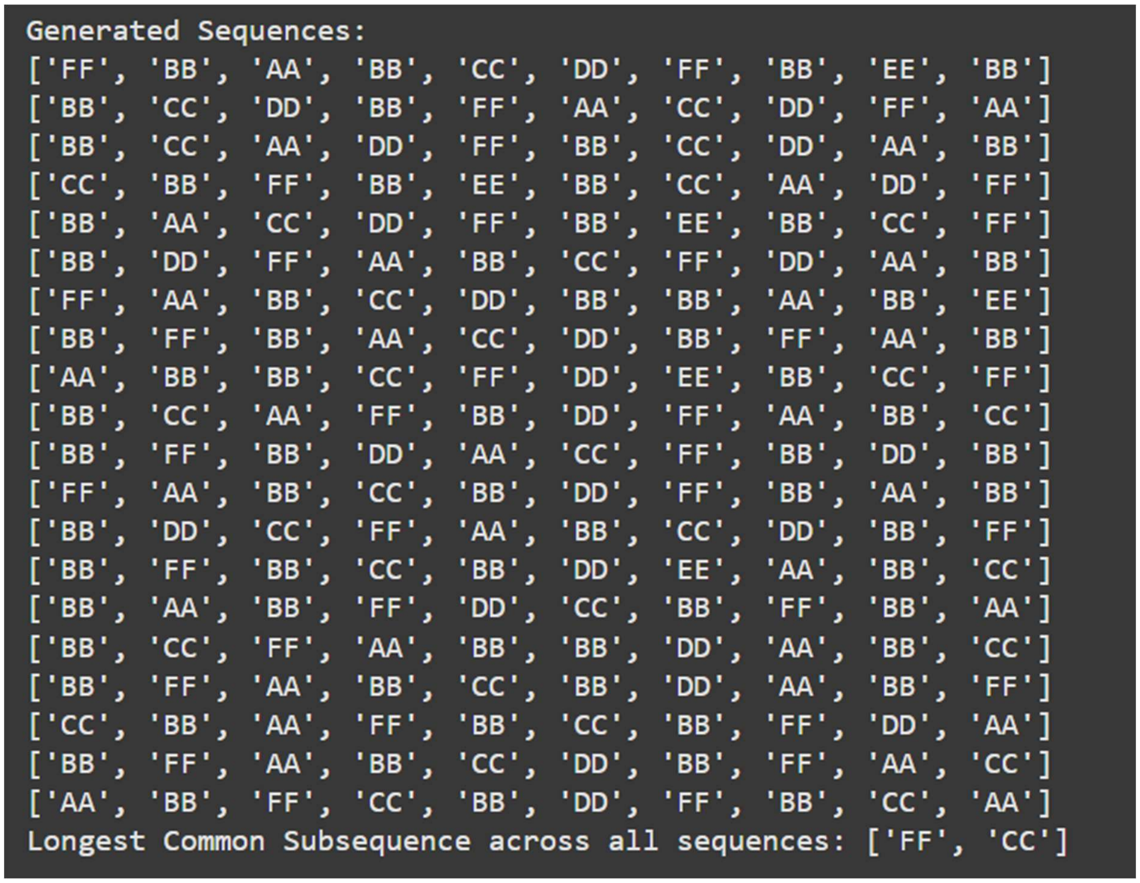
```
Generated Sequences:
['AA', 'BB', 'CC', 'FF', 'DD', 'EE', 'AA', 'BB', 'CC', 'FF']
['BB', 'CC', 'AA', 'EE', 'BB', 'FF', 'AA', 'DD', 'CC', 'BB']
['CC', 'AA', 'BB', 'DD', 'FF', 'BB', 'CC', 'EE', 'BB', 'AA']
['BB', 'DD', 'FF', 'AA', 'BB', 'CC', 'EE', 'BB', 'AA', 'BB']
['DD', 'FF', 'BB', 'CC', 'AA', 'BB', 'CC', 'DD', 'FF', 'AA']
['EE', 'BB', 'FF', 'AA', 'BB', 'CC', 'EE', 'DD', 'FF', 'BB']
['BB', 'AA', 'BB', 'CC', 'DD', 'FF', 'EE', 'BB', 'AA', 'FF']
['CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'BB', 'AA', 'BB', 'CC']
['BB', 'FF', 'DD', 'BB', 'AA', 'CC', 'FF', 'DD', 'BB', 'CC']
['FF', 'CC', 'BB', 'DD', 'AA', 'EE', 'BB', 'AA', 'BB', 'FF']
['BB', 'AA', 'BB', 'CC', 'DD', 'FF', 'BB', 'CC', 'AA', 'BB']
['FF', 'BB', 'CC', 'AA', 'DD', 'EE', 'BB', 'FF', 'BB', 'AA']
['AA', 'BB', 'CC', 'BB', 'FF', 'DD', 'EE', 'BB', 'FF', 'CC']
['BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'CC', 'BB', 'DD', 'AA']
['CC', 'DD', 'AA', 'BB', 'FF', 'DD', 'BB', 'CC', 'FF', 'BB']
['BB', 'FF', 'BB', 'CC', 'DD', 'EE', 'AA', 'BB', 'DD', 'FF']
['BB', 'CC', 'DD', 'AA', 'FF', 'BB', 'CC', 'BB', 'AA', 'FF']
['BB', 'DD', 'FF', 'CC', 'AA', 'BB', 'EE', 'BB', 'DD', 'CC']
['FF', 'AA', 'BB', 'CC', 'BB', 'DD', 'FF', 'AA', 'BB', 'CC']
['BB', 'CC', 'FF', 'AA', 'BB', 'EE', 'BB', 'CC', 'DD', 'FF']
Longest Common Subsequence across all sequences: ['AA', 'BB']
```

- Valid TC 4



```
Generated Sequences:
['BB', 'AA', 'CC', 'DD', 'BB', 'FF', 'BB', 'EE', 'BB', 'CC']
['FF', 'CC', 'BB', 'BB', 'AA', 'EE', 'BB', 'DD', 'FF', 'AA']
['DD', 'AA', 'CC', 'BB', 'FF', 'BB', 'BB', 'CC', 'AA', 'BB']
['BB', 'FF', 'BB', 'CC', 'BB', 'EE', 'FF', 'BB', 'CC', 'DD']
['FF', 'AA', 'BB', 'CC', 'DD', 'BB', 'EE', 'BB', 'FF', 'AA']
['DD', 'FF', 'AA', 'BB', 'BB', 'CC', 'DD', 'FF', 'AA', 'BB']
['BB', 'CC', 'FF', 'AA', 'BB', 'BB', 'DD', 'FF', 'BB', 'CC']
['BB', 'FF', 'BB', 'CC', 'BB', 'EE', 'DD', 'BB', 'FF', 'CC']
['AA', 'BB', 'BB', 'CC', 'FF', 'BB', 'AA', 'DD', 'BB', 'EE']
['FF', 'CC', 'DD', 'AA', 'BB', 'BB', 'CC', 'DD', 'AA', 'BB']
['BB', 'CC', 'DD', 'FF', 'AA', 'BB', 'BB', 'CC', 'AA', 'BB']
['FF', 'BB', 'BB', 'AA', 'DD', 'CC', 'EE', 'BB', 'BB', 'FF']
['AA', 'DD', 'FF', 'BB', 'BB', 'CC', 'EE', 'AA', 'DD', 'BB']
['FF', 'BB', 'CC', 'DD', 'AA', 'BB', 'BB', 'CC', 'FF', 'BB']
['BB', 'AA', 'CC', 'FF', 'BB', 'BB', 'DD', 'BB', 'FF', 'CC']
['BB', 'BB', 'CC', 'DD', 'FF', 'BB', 'AA', 'BB', 'BB', 'CC']
['BB', 'AA', 'BB', 'FF', 'DD', 'CC', 'EE', 'BB', 'FF', 'CC']
['BB', 'FF', 'CC', 'BB', 'AA', 'BB', 'BB', 'CC', 'DD', 'AA']
['AA', 'BB', 'DD', 'FF', 'BB', 'BB', 'CC', 'DD', 'AA', 'BB']
['FF', 'BB', 'CC', 'BB', 'AA', 'BB', 'DD', 'FF', 'BB', 'AA']
Longest Common Subsequence across all sequences: ['CC', 'BB']
```

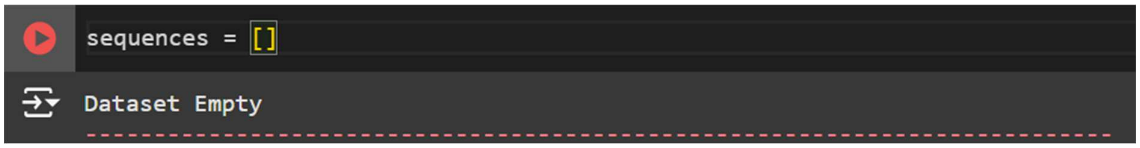
- Valid TC 5



```
Generated Sequences:
['FF', 'BB', 'AA', 'BB', 'CC', 'DD', 'FF', 'BB', 'EE', 'BB']
['BB', 'CC', 'DD', 'BB', 'FF', 'AA', 'CC', 'DD', 'FF', 'AA']
['BB', 'CC', 'AA', 'DD', 'FF', 'BB', 'CC', 'DD', 'AA', 'BB']
['CC', 'BB', 'FF', 'BB', 'EE', 'BB', 'CC', 'AA', 'DD', 'FF']
['BB', 'AA', 'CC', 'DD', 'FF', 'BB', 'EE', 'BB', 'CC', 'FF']
['BB', 'DD', 'FF', 'AA', 'BB', 'CC', 'FF', 'DD', 'AA', 'BB']
['FF', 'AA', 'BB', 'CC', 'DD', 'BB', 'BB', 'AA', 'BB', 'EE']
['BB', 'FF', 'BB', 'AA', 'CC', 'DD', 'BB', 'FF', 'AA', 'BB']
['AA', 'BB', 'BB', 'CC', 'FF', 'DD', 'EE', 'BB', 'CC', 'FF']
['BB', 'CC', 'AA', 'FF', 'BB', 'DD', 'FF', 'AA', 'BB', 'CC']
['BB', 'FF', 'BB', 'DD', 'AA', 'CC', 'FF', 'BB', 'DD', 'BB']
['FF', 'AA', 'BB', 'CC', 'BB', 'DD', 'FF', 'BB', 'AA', 'BB']
['BB', 'DD', 'CC', 'FF', 'AA', 'BB', 'CC', 'DD', 'BB', 'FF']
['BB', 'FF', 'BB', 'CC', 'BB', 'DD', 'EE', 'AA', 'BB', 'CC']
['BB', 'AA', 'BB', 'FF', 'DD', 'CC', 'BB', 'FF', 'BB', 'AA']
['BB', 'CC', 'FF', 'AA', 'BB', 'BB', 'DD', 'AA', 'BB', 'CC']
['BB', 'FF', 'AA', 'BB', 'CC', 'BB', 'DD', 'AA', 'BB', 'FF']
['CC', 'BB', 'AA', 'FF', 'BB', 'CC', 'BB', 'FF', 'DD', 'AA']
['BB', 'FF', 'AA', 'BB', 'CC', 'DD', 'BB', 'FF', 'AA', 'CC']
['AA', 'BB', 'FF', 'CC', 'BB', 'DD', 'FF', 'BB', 'CC', 'AA']
Longest Common Subsequence across all sequences: ['FF', 'CC']
```

NEGATIVE

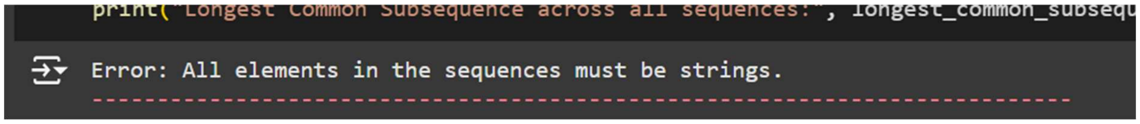
- Empty Dataset



```
sequences = []

Dataset Empty
```

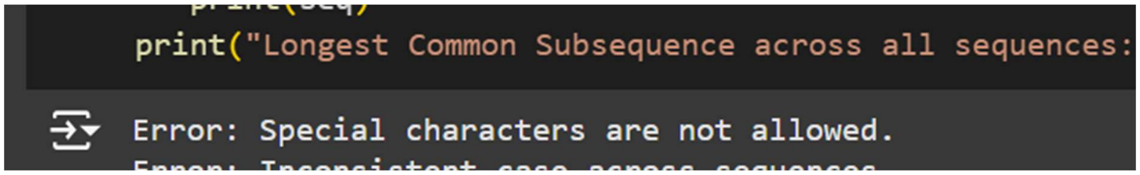
- Non String value in dataset



```
print("Longest Common Subsequence across all sequences:", longest_common_subsequ

Error: All elements in the sequences must be strings.
```

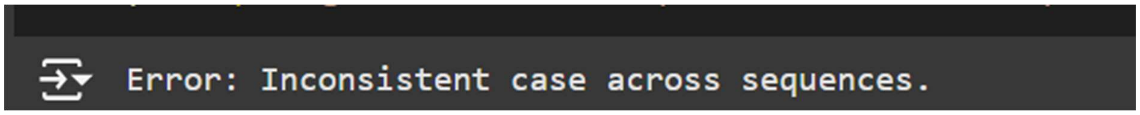
- Special Characters in dataset



```
    print("Longest Common Subsequence across all sequences:

Error: Special characters are not allowed.
```

- Different cases (Uppercase and Lowercase) in Dataset



```
Error: Inconsistent case across sequences.
```

4. **CONCLUSION**

   Hence we implemented recursive LCS approach using memorization for calculating longest common subsequence of 20 sequences consisting of grades of students across different subjects in a year.