Longest Common Subsequence:

Positive test cases:

LCS > positive_test_case_1.csv > data

```
Student ID,Grades
S1,FFCDAAAACDDDCCAABBABBBDDABBBBBAACDDDAAFF
S2,ABAAAABCCDBBFFCCCCCDDDABABFFBBBBAACCFFAB
S3,FFCDFFAAAACDAACDAAABCDAAABABCCABBBDDDDCC
S4,AABBCCCDFFAAABDDBCBBFFCDFFABDDFFBBDDAAFF
S5,DDBCDDABBBCCBCDDBBBCBBBBFFFFBCBCBBCCBCFF
S6,ABFFFFBBAAABCCFFABCDAACCCCFFBBAACCDDABAB
S7,BBCCAABCCCABABCDBCCDDDCCAAAADDFFCDAABBAB
S8,BCDDFFAABCAAABCDAACDCDFFABAABBDDDDCDDDCD
S9,BCBCDDDDDDAAABBCAAAAFFFFCDAAFFAABCBBCCCC
S10,CDBBAABCBCAACCABAACDBCBCABFFAACDABABABCC
S11,ABDDCCDDCCABDDAABCABCDAACCFFDDDDAACDDDBC
S12,BCABCDDDBBABCDBBCCDDBBFFDDAAAABCAAAAAAAA
S13,DDDDCCDDBCCDABFFCCCCABAABBDDABBCBBCDFFDD
S14,DDAACCFFABDDAACCDDCCCDDDAACCCDCDCCCDBCCD
S15,ABDDBCABAAABCDFFBBCDBCBCAAFFCDCDDDBBDDCC
S16,CDAAAAAAFFBCCCAAFFCCDDFFDDBCCCCDCDBBAAAB
S17,DDCCBBAAFFAACDAAABBBAABBBCBBBBDDABBCBCAB
S18,FFAAFFCCCDABFFFFBBAACDFFAABBABABCDFFBBBC
S19,CDCDBBBBCDAAAAAAAACDCDDDDDCCCDABCCAADDBB
S20,AAFFCDAADDAAAABBDDBCBCCDCDDDFFAAFFBCABAB
```

LCS > positive_test_case_2.csv > data

```
Student ID,Grades
S1,BCCCFFBBDDDDCDAADDAACCDDCDDDAABCBBFFAAAA
S2,CDFFABBBABCCBBDDBCBBFFCDFFABFFBBDDABCDCC
S3,FFAAABABCCFFBBAABBCCABABAACDCCCCAAFFAADD
S4,FFCCBBBBAACCCDABABABDDAACDDDFFCDAAAAAAAA
S5,CDCCBCCDAACCBCFFBCFFBCFFABCCCCBCBCCDFFAA
S6,AACDABABBCFFCDFFDDDDDDDDAABBCDCCBBFFBBFF
S7,ABBBABABFFABCCCCBBCCAACCAACDFFCDFFCCDDDD
S8,CDFFDDBCBBABAADDCCABFFFFDDBCFFCDCDFFBBAB
S9,DDABFFBCABCDAADDBBDDABBBDDFFFFFFAAAABCDD
S10,CCCDCCBBDDBBABDDFFDDCDFFDDABAACDDDABBCCD
S11,ABBBABABDDCCABCCBBDDABAACCBCCCCCBBBCCDFF
S12,CDDDBCBBBCDDDDDDCDCCCCCCABABBCCCDDCDCDCD
S13,DDCDBCBBDDCDBBABAABCFFFFBCBBCDDDCCBBDDAA
S14,CCABCCBCCDDDCCAAAAFFCDBCCCCDDDCDCDCDBBCC
S15,AABCCDBBBCCDDDDDFFAABBAABBCDBBCDBCABAABB
S16,AABBBCABABAACCBBBBFFFFCDCDCCDDBBBBBBBBCD
S17,CCBBAACCDDAABBCCFFCDDDCDABCCBBABFFBCFFCD
S18,BBAACDBCCCBCFFCDBCDDDDABCCAABCCDBCFFBBBC
S19,AAAAABCDCCABBCCCFFCCABBCBBBCFFBCAAFFBCBC
S20,CCBBAACDAABCCCFFABAABBBCFFDDDDAACCBBBCAB
```

LCS > positive_test_case_3.csv > data

```
Student ID,Grades
S1,BBBCFFDDCCCCFFFFFFFFCCFFCDBCBBAACDCCFFFF
S2,ABAAABAABBBCABCCCDBCABBBCCFFCCFFCCAABBCD
S3,BBBBABCDFFBBCDAABBCCFFBBBBAACDABBCFFAACC
S4,BBFFAADDDDCDBCCDCDAABBBBCCCDCDAABCDDCDCD
S5,CDBBBCCDBBCDCDBBCCFFABDDCDBCDDBCBBDDBCAB
S6,BBCCABABCCABBBABCCDDFFCDAABBABBBFFFFCCCC
S7,CCDDABCDABAAABBCCCAAAABBBBDDAABBFFBCDDAB
S8,FFDDCCFFCCABCCAABBCDDDDDBBABABFFBCBCFFDD
S9,DDABABABBBCDAABCDDCDAADDABDDCCCCBBCDABAB
S10,CDAACCDDAACDCCABCCFFCDAAFFABBBBBCCDDABFF
S11,CCBCDDBCCDCCABCDAACCCDCCBCFFCDCDAACDCCFF
S12,AAAABBBBAACCFFCCBBBCCCFFFFCCCDDDABBCFFBB
S13,DDDDCCCDABFFBCFFCCBBDDBBABDDDDDDCDCCDDCC
S14,ABFFCDCDBCCDCCAACDCCABFFCCBBABFFCDAACCCC
S15,BCCDCDCDCDCDBBDDCDAABCCDCDCCBBFFBCCCAABB
S16,CCDDBCCCFFBBABCCCDFFABBCCDFFAAAAAABCFFAA
S17,FFCCBBBCAAFFABBCCCAAAACDCDDDBCAACDBBBCBB
S18,BCFFAACCDDBBFFBCCDCCCCCDAABCAADDDDBCCCBB
S19,AACCBBCCDDBCABBBCCBBBCFFCDBBABAACDAAABDD
S20,CCCCCDABABBCCDAAAACDCDABFFDDBCDDDDDDDDD
```

LCS > positive_test_case_4.csv > data

```
Student ID,Grades
S1,CCBBCCBCCCCCBCFFDDFFBCBBDDABBBBCBCCDCDAA
S2,CDAAAACDBCBCFFAAAABBBBCCBCCCFFBCBBCCCDBB
S3,AABCBBBBAAFFCDDDDDABFFDDDDDDCCCDBBABDDAA
S4,BBBCFFBBCDCCFFBCFFABCDABCDDDBCBBABFFAADD
S5,CDABBCAACCDDCDABAADDCCFFFFDDCCCCFFAACCDD
S6,CDCCBCBBCDBCCDDDBBBBCDCCFFBCBBDDCCABBCDD
S7,CCBCCDFFBCBBAABBCCBBDDCCDDCDABCDCCBCAAAA
S8,BBBBCCCDAACCDDDDBBFFDDCDBBBBBBBBAABCAACC
S9,BCAAABFFABBCFFCDCDAABBCDBBCDFFFFBBFFABFF
S10,BCBBDDCCCDBBCCAAAAFFBBABAAFFFFFFBCDDFFAB
S11,ABDDBBDDBBBBBBABBCBCBCAABBCDCDFFBCCDBBFF
S12,ABBCDDCCABCDAADDDDAACCCDDDBCBBCDCCDDBBAA
S13,CCFFCCFFDDCCBBCDCDFFBBCCFFAADDABBBBBCCBC
S14,AACCBBBCBBCDBCFFBCBCBBABBCBCDDAADDCCAACD
S15,ABDDABCCBBABCDDDBCCCDDDDAAFFCDBBCCDDCCBB
S16,FFBBBBBBAAAAFFCCFFCDABAACDBCFFAACDFFFFAB
S17,BBFFAAFFFFCDAACCCCBBDDBCBCFFABCCBBBCABAB
S18,BBBCCDCDABBCDDABBBAABCABCCCCAAABFFABCCCC
S19,AAFFBBDDAABCAACCCDDDBCDDCCCDDDFFAAFFAADD
S20,BBAABCBBCCBBFFDDCDFFAAABFFAACCBCBCAACDCC
```

LCS > positive_test_case_5.csv > data

```
Student ID,Grades
S1,CCABBBCCBBFFAABCABCCBBBBBCCDBCDDBCCCBBCD
S2,BCBCDDBCBBAAABBCCCFFBBABBBFFAAFFDDABCCFF
S3,BBFFABAAAADDAAFFCDBCFFBBAACCAABBFFAAABFF
S4,ABFFAABBAAABCDBBCDABABAAABFFCDAABCBBABBB
S5,BCFFBBABABBCCCAACDFFFFBCCDDDABDDFFDDCCCC
S6,CDCDCCABFFDDABBCAACCABABBBFFCDFFCDDDCDCD
S7,BCBBBBFFBBBCBBFFCDFFBBAAAACCABDDABBBFFCC
S8,CCAAAADDBBBCDDCDDDBBDDCDFFCDABFFAACDCDCC
S9,CDDDDDCCDDDDABCDCDBCBBBCCCABBBBCABAAABBC
S10,BBABBBBCAACCBCABAACDABCDDDABBCCDAADDBBAA
S11,DDAAABCCBBCDABDDDDABCCDDBBFFCCAADDCDBBCD
S12,ABDDFFBCABBBAABBCDFFBCAACCFFABCCCDBBCCBB
S13,CCDDAAFFFFDDBBCCCDBBFFBCCDFFBCFFDDABCDFF
S14,BCCDDDBCCDAABBCDAADDABABBCCCABBCCCBBAAFF
S15,BCCCBBABFFCDAADDBBDDCCBBFFBBFFCCDDCDCDCC
S16,AACCDDABBCAACCBCABCDABFFAACDCCAACCCCFFCD
S17,DDFFABAADDABBCCCBBAAABFFAAAADDCCBCCDAABB
S18,ABBCCCFFCDBCDDCDFFFFBBAAFFBCCCDDBBFFCDCC
S19,CCAAAACDBBCCCDCDAAAABCDDBCBBAAAABCDDFFCC
S20,BCFFDDCDFFAAAACDBCABCCDDBBCCCCCCCDDDABFF
```

```
Positive Test Cases:
Longest Common Subsequence of Grades for All Students in Test Case 1: ABB
Longest Common Subsequence of Grades for All Students in Test Case 2: CCBB
Longest Common Subsequence of Grades for All Students in Test Case 3: CCCCCCC
Longest Common Subsequence of Grades for All Students in Test Case 4: BCCCC
Longest Common Subsequence of Grades for All Students in Test Case 5: BBBBB
venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 %
```

Negative test cases:

LCS > negative_test_case_empty.csv > data

```
Student ID,Grades
S1,
S2,
S3,
S4,
S5,
S6,
S7,
S8,
S9,
S10,
S11,
S12,
S13,
S14,
S15,
S16,
S17,
S18,
S19,
S20,

```

LCS > negative_test_case_invalid_grade_code.csv > data

```
Student ID,Grades
S1,AABBDDBBFFAAAAAAFFFFCCABDDBCDDDDABAACDZZ
S2,CCBBFFABABFFCDBBDDABBBAAFFBCCDDDFFCCCCZZ
S3,AAABCCCCBBDDCCCCABCCCCABDDCDBCBCCCFFABZZ
S4,BCDDAABCCCAAAABBFFAAFFCDDDBBCCCDBCCDABZZ
S5,ABABDDABDDDDBBBCBBCCCCAABBAADDABCCABABZZ
S6,DDFFFFBBBBABBCAAAAABCDBBDDBBDDAABCFFCCZZ
S7,FFCDABABCCABDDBCABCCCCBBAADDFFBCABDDBCZZ
S8,CDBBCDCCCDAAABCDAABCABAADDFFDDCCBCBCBCZZ
S9,AABCFFFFAACDCDCDFFBCCCDDBBABDDBCABAACCZZ
S10,BCCCABABBCABDDCCFFDDBBCDBCFFBCAAFFAACDZZ
S11,AAAADDFFABBBCDBBCDDDDDAADDBCCDFFCDDDFFZZ
S12,ABFFBCBBCCCCAACDCDAAABFFCDAAFFABBCABCCZZ
S13,CDCCBBBCBBBBABCCABDDDDFFBBBCFFFFCDABCCZZ
S14,BCAACCBBDDABCCCDCCABAAFFCCCCCDBBFFBBFFZZ
S15,AACDFFCDABDDBBCCABCCBBDDABBBCDCDFFCDDDZZ
S16,DDAAFFBBCCCDFFCCABCDFFCDFFFFCDFFAADDCDZZ
S17,AAFFDDABCDBCCCAAAADDCDBCAAFFDDABABABCCZZ
S18,BCBCAACCAABBFFBBCDABCCFFABDDFFBCFFCDBCZZ
S19,FFCDBCDDDDCDAABBBBCDABBBDDCDCCABCCBCDDZZ
S20,CDDDCDABBCBCCCCDCCFFBBAABBDDBBCCCCDDABZZ

```

LCS > negative_test_case_numbers_present.csv > data

```
Student ID,Grades
S1,AAABDDBCCCDDCDBCAACDDDBBCDFFCCBCABBCAB1
S2,AABCBCFFCCBBABCDBCCDABCCCCABBCABCDABBC1
S3,AABBDDCDDDCCBCABABCDDDCCCDFFDDBCBBDDAB1
S4,AABBFFBCCCFFDDFFABBBBCBBBBFFFFBBFFCCCC1
S5,AABCBCBBABABCDFFCDCCABAACCCCDDABBCAABB1
S6,AAABFFCDCDFFBCABCCBCAABCBCBCFFBBCDCDBC1
S7,AAFFABBBABDDCDBBABFFABDDCDCCDDABCDCCBB1
S8,AAFFFFABCCDDCDCCBBBCFFABBCCDFFBCBBBBBB1
S9,AABCAADDBCAADDBBCDBCBBABABAABBCDAACDCD1
S10,AAAAABAAFFFFABDDFFABFFABCDABCDFFABBBAB1
S11,AAAABCBBABFFBBABDDBBCCABBCDDDDCDAAABBC1
S12,AADDABCDFFCDBBCDCCABBCCDDDABFFBBDDCDFF1
S13,AAAAAACDABCDCDFFABAAFFABBBAAAAABDDCCDD1
S14,AAABCCCCAADDCCFFDDFFCCCCBCCCCDBBCDAABC1
S15,AAAABBCCBCDDFFCDAADDCCCDBBDDCCAABCBBCD1
S16,AABCDDBCCDBCCDCCAAAADDDDAACCDDCDCDCCCC1
S17,AABBAAFFCDAAABFFCCDDABBCBBCCBCFFCCAACC1
S18,AACDABAACDABBBABCDFFCDAAAACCBCCCAACCAA1
S19,AAAABCCCCCABCCDDABABBBAADDAADDBCFFBCCC1
S20,AAAAABAACDCCCCCDABDDCDBCCCCCFFDDFFABAB1
```

LCS > negative_test_case_same_grade.csv > data

```
Student ID,Grades
S1,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S2,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S3,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S4,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S5,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S6,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S7,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S8,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S9,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S10,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S11,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S12,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S13,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S14,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S15,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S16,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S17,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S18,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S19,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
S20,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

LCS > negative_test_case_special_characters.csv > data

```
Student ID,Grades
S1,AACCDDFFCDABBCCCDDCCABCCAAFFABCCCCAABB%
S2,AADDBBCCFFFFCDDDBBDDAACDFFABFFABAAFFBC%
S3,AABBBCBBABAACCCDABCCCDBCBBCDABCCDDABDD%
S4,AABCDDCDCCCDBCFFCDBCDDDDAABCCCDDAAAAAB%
S5,AABCAADDCCABBCCCAACDFFCCABBCABCDAAABBB%
S6,AABCABABCDCDCCFFDDBCABCCBCCDDDCCABCCAA%
S7,AACDCDCCDDCDBBDDDDAABBCCDDAADDDDBCFFFF%
S8,AABCBBAABCCDABAAABAABCCDCCDDFFBCBCBBCC%
S9,AAFFABAACDAAABDDAAABBCDDDDAAAACDBCBBCC%
S10,AAABCDABCCCDDDFFBBBCCCABDDBCABFFCDFFCD%
S11,AACDFFABAAAABBCCCCBCBCABBCCDFFCCFFDDFF%
S12,AAAAFFCDFFBCBCCCABCDFFFFABBBDDFFBCCCAA%
S13,AAFFAAABAACCFFFFBBDDCDCDBBABCDAAFFBCFF%
S14,AAAADDBBBCDDCCCDBCFFFFAAFFAACDBCCDCCBB%
S15,AAAAFFFFABABCDFFFFFFABBCCDBCCDBCCDCCCD%
S16,AADDBBABDDDDDDFFBCCDBCBBCDBBCCAADDBCFF%
S17,AABBFFBCBBAAFFAACCBCCCCDABABAAFFBCCDFF%
S18,AAFFABAAABDDFFBBAAAAAACCAABCFFBCBBFFBB%
S19,AABBBCBBCCCDABCCBCCCAACCDDBCFFBBCDBBAA%
S20,AACCFFFFCDCDAAFFFFCDCDABCCDDCCABBCCCCC%
```

```
Negative Test Cases:
Test Case 1: Error for student S1: Invalid grade sequence: AABBDDBBFFAAAAAAFFFFCCABDDBCDDDDABAACDZZ. Special characters or invalid characters found.
Test Case 2: Student S1 - Missing or invalid grade data.
Test Case 2: Error for student S1: Invalid grade sequence: nan. Grade data should be a valid string.
Test Case 3: Error for student S1: Invalid grade sequence: AACCDDFFCDABBCCCDDCCABCCAAFFABCCCCAABB%. Special characters or invalid characters found.
Test Case 4: Error for student S1: Invalid grade sequence: AAABDDBCCCDDCDBCAACDDDBBCDFFCCBCABBCAB1. Numbers found in the sequence.
Test Case 5: Student S1 - Grades are valid
Test Case 5: All students have the same grade: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 %
```

Matrix chain multiplication:

Positive test cases:

```
matrices_1 = [4, 7, 4, 7, 4, 7]  # [4x7, 4x7, 4x7]
matrices_2 = [3, 7, 3, 7, 3, 7, 3, 7]  # [3x7, 3x7, 3x7, 3x7]
matrices_3 = [5, 7, 5, 7, 5, 7, 5, 7, 5, 7]  # [5x7, 5x7, 5x7, 5x7, 5x7]
matrices_4 = [6, 7, 6, 7, 6, 7, 6, 7, 6, 7, 6, 7]  # [6x7, 6x7, 6x7, 6x7, 6x7, 6x7]
matrices_5 = [3, 7, 3, 7]  # [3x7, 3x7]
```

```
"/Users/mihirkatakdhond/Downloads/daa lab 6/venv/bin/python" "/Users/mihirkatakdhond/Downloads/daa lab 6/mcm.py"
venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 % "/Users/mihirkatakdhond/Downloads/daa lab 6/venv/bin/python" "/Users/mihirkatakdhond/Downloads/daa lab 6/mcm.py"
Test Case 1:
Minimum number of scalar multiplications: 400
Optimal multiplication order: (((A1 x A2) x (A3 x A4)) x A5)

Test Case 2:
Minimum number of scalar multiplications: 306
Optimal multiplication order: (((A1 x A2) x ((A3 x A4) x (A5 x A6))) x A7)

Test Case 3:
Minimum number of scalar multiplications: 1250
Optimal multiplication order: (((A1 x A2) x ((A3 x A4) x ((A5 x A6) x (A7 x A8)))) x A9)

Test Case 4:
Minimum number of scalar multiplications: 2376
Optimal multiplication order: (((A1 x A2) x ((A3 x A4) x ((A5 x A6) x ((A7 x A8) x (A9 x A10))))) x A11)

Test Case 5:
Minimum number of scalar multiplications: 126
Optimal multiplication order: ((A1 x A2) x A3)

venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 %
```

Negative test cases:

```python
# Negative Test Case 1: Empty List (No matrices)
def test_empty_list():
    matrices = []


# Negative Test Case 2: Single matrix (only one matrix provided)
def test_single_matrix():
    matrices = [5, 7]  # Only one matrix


# Negative Test Case 3: Incompatible matrix dimensions
def test_incompatible_dimensions():
    matrices = [5, 7, 6, 8, 7, 9]  # Incompatible matrices, cannot multiply


# Negative Test Case 4: Invalid matrix dimensions (negative values)
def test_invalid_dimensions():
    matrices = [5, 7, -6, 8, 7, 9]  # One matrix has negative dimension


# Negative Test Case 5: Non-square matrices (unexpected)
def test_non_square_matrices():
    matrices = [5, 7, 7, 5, 5, 7]  # Matrices with different row-column numbers
```

```
"/Users/mihirkatakdhond/Downloads/daa lab 6/venv/bin/python" "/Users/mihirkatakdhond/Downloads/daa lab 6/mcm2.py"
venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 % "/Users/mihirkatakdhond/Downloads/daa lab 6/venv/bin/python" "/Users/mihirkatakdhond/Downloads/daa lab 6/mcm2.py"
Test Case 1: Empty List
Cost: -1, Order: Matrix dimension array length should be N-1

Test Case 2: Single Matrix
Cost: 0, Order: A1

Test Case 3: Incompatible Matrix Dimensions
Cost: -1, Order: Incompatible matrix dimensions for multiplication

Test Case 4: Invalid Matrix Dimensions
Cost: -1, Order: Matrix dimensions must be positive

Test Case 5: Non-square Matrices
Cost: -1, Order: Incompatible matrix dimensions for multiplication

venvmihirkatakdhond@Mihirs-MacBook-Air daa lab 6 %
```

1. Single Responsibility Principle (SRP)

A class should have only one reason to change, meaning it should have only one job or responsibility.

Problem: A class should have only one responsibility. In the first example, the `Employee` class does both salary calculation and report generation, violating SRP.

Code:

```
class Employee:
    def __init__(self, name):
        self.name = name

    def calculate_salary(self):
        return 50000

class ReportGenerator:
    @staticmethod
    def generate_report(employee):
        return f"Report for {employee.name}"
```

Explanation: The `Employee` class now only handles employee-related tasks (salary calculation), while the `ReportGenerator` class is responsible for generating reports. Each class has a single responsibility.

---

2. Open/Closed Principle (OCP)

Software entities (classes, modules, functions) should be open for extension but closed for modification.

Problem: Classes should be open for extension (i.e., new functionality) but closed for modification (i.e., not modifying existing code). In the example below, we don't modify the `Shape` class to add new shapes, we extend it.

Code:

```
class Shape:

    def area(self):
        pass
```

```
class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2
```

Explanation: The `Shape` class is not modified. Instead, new shapes like `Rectangle` and `Circle` extend `Shape` and implement the `area` method, adhering to the Open/Closed principle.

---

3. Liskov Substitution Principle (LSP)

Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.

Problem: Subtypes should be able to replace their base types without affecting the functionality. Here, the `Penguin` class violates the LSP because it cannot fly, unlike its superclass `Bird`.

Code:

```
class Bird:
    def fly(self):
        pass

class Sparrow(Bird):
    def fly(self):
        print("Flying")

class Penguin(Bird):
```

```
    def fly(self):  # This method is intentionally empty or throws an
exception
        pass
```

Explanation: The `Penguin` class doesn't implement flying because penguins can't fly. In a better design, we'd avoid such a situation by having `Bird` only define methods that all birds can implement (e.g., walking or swimming), or we could create separate classes like `FlyingBird` and `NonFlyingBird` to maintain the LSP.

---

4. Interface Segregation Principle (ISP)

Clients should not be forced to depend on interfaces they do not use.

Problem: Clients should not be forced to implement interfaces they don't use. In the case of a multifunction printer, both `Printer` and `Scanner` are separate interfaces that can be used individually, so no class is forced to implement both if it doesn't need to.

Code:

```
class Printer:
    def print(self):
        pass

class Scanner:
    def scan(self):
        pass

class MultiFunctionPrinter(Printer, Scanner):
    def print(self):
        pass

    def scan(self):
        pass
```

Explanation: The `Printer` and `Scanner` interfaces are separated. If a device is only a printer, it will implement just the `Printer` interface. If it can scan as well, it implements both `Printer` and `Scanner`. This avoids forcing a device to implement methods it doesn't need, adhering to the Interface Segregation Principle.

---

5. Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

Problem: High-level modules should not depend on low-level modules. Both should depend on abstractions (interfaces). In this case, `UserService` depends on an abstract `Database` class instead of directly on a specific implementation like `MySQLDatabase`.

Code:

```
class Database:
    def connect(self):
        pass

class MySQLDatabase(Database):
    def connect(self):
        print("Connecting to MySQL")

class UserService:
    def __init__(self, db: Database):
        self.db = db

    def save_user(self, user):
        self.db.connect()
        print(f"Saving user {user}")
```

Explanation: The `UserService` class now depends on the abstract `Database` class, not a specific implementation like `MySQLDatabase`. This allows the `UserService` to work with any database that implements the `Database` interface, making the code more flexible and decoupled. The `MySQLDatabase` class is just one possible implementation of the `Database` interface, so the `UserService` can use different types of databases without modification.