MIU System

This problem is a String (containing all Upper case Letters) manipulating problem. You will implement six methods in the MIU\_System class. (Remember, you do **NOT** need to correctly implement all six methods to earn some points. A second reminder, each method you implement correctly will earn you points.)

The MIU\_System class has a constructor with a single String parameter that is copied to the instance variable seed. The String passed to the MIU\_System constructor will always contain a single M as the first element of the String and is followed by the a mix if I and Us. That is, seed.indexOf(“M”) == 0 and seed.substring(1).indexOf("M") == -1. The six methods in this problem all return modified version of the instance variable seed. Please note that the instance variable is not to be modified by any these six methods.

The first method to implement is the doubleAfterM method. This method returns a String that doubles the entire seed following the “M”. Remember, seed must not be modified.

The following code shows the results of the doubleAfterM method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MI"); |  |
| m.doubleAfterM(); | "MII" |

The following code shows the results of the doubleAfterM method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MUIIU"); |  |
| m.doubleAfterM(); | " MUIIUUIIU" |

The second method to implement is the endsWithI method. This method returns a String that adds a U to the end of the seed only if seed ends with an I. If seed does **not** end with an I, return seed. Remember, seed must not be modified.

The following code shows the results of the endsWithI method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MII") |  |
| m.endsWithI(); | " MIIU" |

The following code shows the results of the endsWithI method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MIU") |  |
| m.endsWithI(); | " MIU" |

The third method to implement is the trade3IsForSingleU method. This method returns a String that replaces the first occurrence of three I’s (III) with a U. If seed does **not** contain III, return seed. Remember, seed must not be modified.

The following code shows the results of the trade3IsForSingleU method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MIIIUIII"); |  |
| m.trade3IsForSingleU(); | "MUUIII" |

The fourth method to implement is the remove2Us method. This method returns a String that removes all occurrences of two adjacent U’s (UU). If seed does **not** contain UU, return seed. Remember, seed must not be modified.

The following code shows the results of the remove2Us method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MUIUUIUU"); |  |
| m.remove2Us(); | "MUII" |

The following code shows the results of the remove2Us method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MUUUIUUUU"); |  |
| m.remove2Us(); | "MUI" |

The fifth method to implement is the isPossible(String target) method. This method returns true if exactly one of the previous four methods (endsWithI, doubleAfterM, trade3IsForSingleU, remove2Us) return a String that matches the parameter target. And isPossible returns false otherwise. Remember, seed must not be modified.

The following code shows the results of the isPossible method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MI"); |  |
| m.isPossible("MII"); | true |
| m.isPossible("MIU"); | true |
| m.isPossible("MUI"); | false |
| m.isPossible("MIII"); | false |

The sixth method to implement is the minNumModifications(String target) method. This method returns the minimum number of the four MIU\_System method calls required to return a String that matches the parameter target. Remember, seed must not be modified.

* You may assume, 0 <= minNumModifications <=10. That is, it will always be possible to find a sequence of (10 or fewer) method calls that will return a String matching target.
* If target.equals(seed) == true, return 0;

The following code shows the results of the minNumModifications method.

|  |  |
| --- | --- |
| The following code | Returns |
| MIU\_System m = new MIU\_System("MI"); |  |
| m.minNumModifications("MII")); | 1 |
| m.minNumModifications("MUI")); | 3 |
| m.minNumModifications("MIIII")); | 2 |
| m.minNumModifications("MIIIIIIII")); | 3 |
| m.minNumModifications("MUIIIII")); | 4 |
| m.minNumModifications("MUUII")); | 5 |
| m.minNumModifications("MUUIIU")); | 6 |
| m.minNumModifications("MUUIIUUUIIU")); | 7 |
| m.minNumModifications("MIIUIIU")); | 3 |