Instructions:

Read the description of the question in the following. Complete the given partial code, and submit the required .java files to wattle for marking.

You are allowed to define additional methods in your implementation. But you cannot alter the signatures of the given methods and the package structures of the given classes. There should be no package name in all .java files. Please check your code to ensure that it does not violate submission guidelines before submission.Otherwise, you will receive penalty for any violation.

Q1: (20 Marks)

This question extends a red-black tree to a three-color tree data structure.There are the following properties for a three-color tree:

1. Each node is either PINK, PURPLE or MAGENTA.
2. Root and every leaf (NULL pointer) are PINK.
3. Each path from Root-to-Leaf has the same number of PINK nodes.
4. A MAGENTA node cannot have a MAGENTA child, and must at least has a PURPLE child.
5. A PURPLE node must have only PINK children.

Please implement the checking functions testProp1, testProp2, testProp3, testProp4 to check if properties (1), (2), (3), (4) hold. Hint: testProp5() has been already implemented in the partial code. You are allowed to define additional methods in your implementation. But you cannot alter the signatures of the given methods and the package structures of the given classes. Please upload the ThreeColourTree.java file to wattle for marking.

Q2: (20 Marks) You are given a java class called Something, which has a method called `someMethod`. Please implement a minimum number of test cases for testing someMethod that are **branch complete** within `someMethod`. Write your test case(s) in test() method in `SomethingTest.java`. You cannot alter the signatures of the given methods and the package structures of the given classes.

Please upload the `SomethingTest.java` file to wattle for marking.

Q3. In this question, you will implement a simple database in XML supporting a simple SQL command INSERT.

The database is a Customer database. A table has multiple records, where a record represents a row in the table. The data in the same column of a table are associated with the same field name as the records. Each record has a unique numeric ID. The data can only include alphabet, numbers and spaces.

Here is an example of the database:

```
ID | Name          | Address    | City     | Postcode | Country
---------------------------------------------------------------
 1 | 'J Doe'       |'1 Main St'|'Sydney'|    '2000'    |'Australia'
```

Consider the following SQL command:

The SQL INSERT command aims to create a new record in a table and generate a new ID for the new record. It has the following format:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

```
Example:
```

```
INSERT INTO Customers (ID, Name, Address, City, Postcode,
Country)
VALUES (1, 'J Doe', '1 Main St', 'Sydney', '2000', 'Australia');
```

<Command> := INSERT INTO <table_name> (<columns>) VALUES <values>
<table_name> := <str>
<columns> := <var>, <columns> | <var>
<values> := '<str>', '<num>', <values> | '<str>', '<num>'

where <num> is an integer literal,<var> and <table_name> are string literals with only alphabet, <str> is a string literal with only alphabet, numbers and spaces.

Tasks:

Implement the following tasks:

1. (10 Marks) Implement `load() and insert() in XMLTable.java. save() is already given.`

   Each customer must have an ID value, but may not have all the following column values (e.g. Name, Address, City, Postcode, Country). Please see test cases in XMLTableTest.java.

2. (10 Marks) Implement next() method in Tokeniser.java to extract the SQL commands as Tokens as follows

a. Token 1:
originalTokenStr: INSERT INTO table_name (column1, column2, column3, ...)
type: INSERT_INTO
value: table_name (column1, column2, column3, ...)

b. Token 2:
originalTokenStr: VALUES (value1, value2, value3, ...)
type: VALUES
value: (value1, value2, value3, ...)

Note that some brackets in the SQL commands may be missing. Please return null if some brackets are missing. Please see the test cases in TokeniserTest.java

3. (10 Marks) Implement parseExp() in Parser.java to extract the columns and values from tokens and execute the SQL command to insert new customers. Do not insert customers if the following errors are found:
    a. some brackets are missing
    b. some column names are wrong

   Please see the column names in Customer.java file and test cases in ParserTest.java

For each task, we provided you with some JUnit tests, whose file name ends with "[...]Test.java". Please note that when marking, we might use different test cases.

Please upload the following files to Wattle for marking: `XMLTable.java`, `Tokeniser.java and Parser.java`.