

2006 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. This question involves reasoning about the code from the Marine Biology Simulation case study. A copy of the code is provided as part of this exam.

Consider using the `BoundedEnv` class from the Marine Biology Simulation case study to model a game board. In this implementation of the `Environment` interface, each location has at most **four** neighbors. Those neighbors are determined by the `Environment` method `neighborsOf`.

DropGame is a two-player game that is played on a rectangular board. The players — designated as BLACK and WHITE — alternate, taking turns dropping a colored piece in a column. A dropped piece will fall down the chosen column until it comes to rest in the empty location with the largest row index. If the location for the **newly dropped** piece has **three** neighbors that match its color, the player that dropped this piece wins the game.

The diagram below shows a sample game board on which several moves have been made.

| | | | | | | | | | |
|------|---|-------|---|---|---|---|---|------|--|
| | | North | | | | | | | |
| | | 0 | 1 | 2 | 3 | 4 | 5 | | |
| West | 0 | ● | | | | | | East | |
| | 1 | ● | ● | | ○ | | | | |
| | 2 | ● | ○ | | ○ | | ● | | |
| | 3 | ○ | ○ | ○ | ● | | ● | | |
| | | South | | | | | | | |

The following chart shows where a piece dropped in each column would land on this board.

| Column | Location for Piece Dropped in the Column |
|--------|--|
| 0 | No piece can be placed, since the column is full |
| 1 | (0, 1) |
| 2 | (2, 2) |
| 3 | (0, 3) |
| 4 | (3, 4) |
| 5 | (1, 5) |

Note that a WHITE piece dropped in column 2 would land in the shaded cell at location (2, 2) and result in a win for WHITE because the three neighboring locations — (2, 1), (3, 2), and (2, 3) — contain WHITE pieces. This move is the only available winning move on the above game board. Note that a BLACK piece dropped in column 1 would land in location (0, 1) and not result in a win because the neighboring location (0, 2) does not contain a BLACK piece.

2006 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

The `Piece` class implements the `Locatable` interface and is defined as follows.

```
public class Piece implements Locatable
{
    // returns location of this Piece
    public Location location()
    { /* implementation not shown */ }

    // returns color of this Piece
    public Color color()
    { /* implementation not shown */ }

    // There may be fields, constructors, and methods that are not shown.
}
```

An incomplete definition of the `DropGame` class is shown below. The class contains a private instance variable `theEnv` to refer to the `Environment` that represents the game board. Players will add `Piece` objects to this environment as they take turns. You will implement two methods for the `DropGame` class.

```
public class DropGame
{
    private Environment theEnv; // contains Piece objects

    // returns null if no empty locations in column;
    // otherwise, returns the empty location with the
    // largest row index within the specified column;
    // precondition: 0 <= column < theEnv.numCols()
    public Location dropLocationForColumn(int column)
    { /* to be implemented in part (a) */ }

    // returns true if dropping a piece of the given color into the
    // specified column matches color with three neighbors;
    // otherwise, returns false
    // precondition: 0 <= column < theEnv.numCols()
    public boolean dropMatchesNeighbors(int column, Color pieceColor)
    { /* to be implemented in part (b) */ }

    // There may be fields, constructors, and methods that are not shown.
}
```

2006 AP[®] COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (a) Write the `DropGame` method `dropLocationForColumn`, which returns the resulting `Location` for a piece dropped into the specified column. If there are no empty locations in the column, the method should return `null`. Otherwise, of the empty locations in the column, the location with the largest row index should be returned.

In writing `dropLocationForColumn`, you may use any methods defined in the `DropGame` class or accessible methods of the case study classes.

Complete method `dropLocationForColumn` below.

```
// returns null if no empty locations in column;  
// otherwise, returns the empty location with the  
// largest row index within the specified column;  
// precondition: 0 <= column < theEnv.numCols()  
public Location dropLocationForColumn(int column)
```

- (b) Write the `DropGame` method `dropMatchesNeighbors`, which returns `true` if dropping a piece of a given color into a specific column will match the color of three of its neighbors. The location to be checked for matches with its neighbors is the location identified by method `dropLocationForColumn`. If there are no empty locations in the column, `dropMatchesNeighbors` returns `false`.

In writing `dropMatchesNeighbors`, you may assume that `dropLocationForColumn` works as specified regardless of what you wrote in part (a).

Complete method `dropMatchesNeighbors` below.

```
// returns true if dropping a piece of the given color into the  
// specified column matches color with three neighbors;  
// otherwise, returns false  
// precondition: 0 <= column < theEnv.numCols()  
public boolean dropMatchesNeighbors(int column, Color pieceColor)
```

END OF EXAM