

## 2003 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

4. This question involves reasoning about the code from the Marine Biology Case Study. Please reference a copy of the code.

The marine biologists want to study a species of fish that eats algae. Any position in the environment grid can contain zero or more units of algae. If there is any algae at a fish's location, the fish eats one unit of algae and does not move; otherwise, the fish does not eat. If this is the third consecutive step in which the fish has not eaten, then the fish dies and is removed from the environment. If the fish does not eat and does not die, it moves to a position among its empty neighbors that contains the most algae.

We represent the algae by adding a matrix of integers to the private data of the `Environment` class. This matrix is the same size as `myWorld`, and each entry represents the number of units of algae at that location. We add two public member functions to the `Environment` class, as well as modifying the `Environment` constructor to initialize `myAlgae`.

```
//      Added these functions to the Environment class

public int NumAlgaeAt(Location loc)
//precondition:  loc is a valid Location in the environment
//postcondition: returns the number of units of algae at loc

public void RemoveAlgae(Location loc, int numUnits)
//precondition:  algae at Location loc exceeds numUnits
//postcondition: algae at Location loc has been reduced by numUnits

//      Added this data member to the Environment class
private int[][] myAlgae;
```

## 2003 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

We modify the `Fish` class by adding a private data member to keep track of how long since the fish ate any algae. We also add public member function `Go` that encapsulates all of the actions of a fish for one step of the simulation, and we modify the `move` function so that the fish moves to the position among its empty neighbors that has the most algae. In order to do this we add private member function `MostAlgae` to the `Fish` class.

```
//      Added these functions to the Fish class

    public void Go()
    //precondition:  this Fish is stored in environment() at location()
    //postcondition: if there was algae at location(), this Fish ate and
    //              one unit of algae has been removed from
    //              location(); otherwise, if this was the third
    //              consecutive step that this Fish did not eat,
    //              then this fish has been removed from
    //              environment(); otherwise, this Fish moved.
    //              myStepsSinceFed has been updated.

    private Location MostAlgae(ArrayList nbrs)
    //precondition:  nbrs.size() > 0, this Fish is in Environment
    //              environment()

    //postcondition: returns a Location from nbrs that contains the
    //              most algae.

//      Modified these functions in the Fish class (changes are in bold)
private void move()
//postcondition:  this Fish tried to move to the adjacent space to
//              location() with the greatest number of algae in
//              environment().

//      Added this data member to the Fish class

    private int myStepsSinceFed; // steps since this fish last ate
```

- (a) Write the `Environment` member function `NumAlgaeAt`, which is described as follows. `NumAlgaeAt` should return the number of units of algae at `Location loc`.

Complete the function `NumAlgaeAt` below.

```
public int NumAlgaeAt(Location loc)
//precondition:  loc is a valid Location in the environment
//postcondition: returns the number of units of algae at loc
{
    return myAlgae[loc.row()][loc.col()];
}
```

## 2003 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (b) Write the `Fish` member function `MostAlgae`, which is described as follows. `MostAlgae` should return a `Location` from `nbrs` that contains the most algae. If more than one position contains the maximum amount, any of those positions may be returned.

In writing `MostAlgae`, you may use any of the `Environment` public member functions, including `NumAlgaeAt`. Assume that `NumAlgaeAt` works as specified, regardless of what you wrote in part (a).

Complete the function `MostAlgae` below.

```
private Location MostAlgae(ArrayList nbrs)
//precondition:  nbrs.size() > 0, this Fish is in Environment
//              environment()
//postcondition: returns a Location from nbrs that contains the
//              most algae.
{
    int mostPos=0;

    for(int i=1; i<nbrs.size(); i++)
    {
        int here=environment().NumAlgaeAt((Location)nbrs.get(i));
        int old=environment().NumAlgaeAt((Location)nbrs.get(mostPos));
        if(here>old)
            mostPos=i;
    }
    return (Location)nbrs.get(mostPos);
}
```

## 2003 AP® COMPUTER SCIENCE A FREE-RESPONSE QUESTIONS

- (c) Write the `Fish` member function `Go`, which is described as follows. If there is algae at the fish's current `Location`, the fish should eat one unit of algae and not move. If there is no algae and this is the third consecutive step in which the fish has not eaten, `Go` will cause the fish to die by calling the `Environment` member function `RemoveFish`. If the fish does not eat and does not die, then the fish should move to a neighboring location with the most algae. `Go` should update the state of the `Environment` and the state of the `Fish` appropriately.

In writing `Go`, you may use any member function from the Marine Biology Case Study, including those added at the beginning of the question. Assume that the `Fish` member function `move` has been modified to work correctly and that the `Environment` member function `NumAlgaeAt` and the `Fish` member function `MostAlgae` work as specified, regardless of what you wrote in parts (a) and (b).

Complete function `Act` below.

```
public void Go()
//precondition:  this Fish is stored in environment() at location()
//postcondition: if there was algae at location(), this Fish at and
//              one unit of algae has been removed from
//              location(); otherwise, if this was the third
//              consecutive step that this Fish did not eat,
//              then this fish has been removed from
//              environment(); otherwise, this Fish moved.
//              myStepsSinceFed has been updated.
{

    if(environment().NumAlgaeAt(location())>0)
    {
        environment().RemoveAlgae(location(),1);
        myStepsSinceFed=0;
    }
    else if(myStepsSinceFed>=3)
    {
        environment().remove(this);
        return;
    }
    else
    {
        move();
        myStepsSinceFed++;
    }
    environment().recordMove(this,location());
}
```