

AP[®] Computer Science A 2003 Sample Student Responses

The materials included in these files are intended for use by AP teachers for course and exam preparation; permission for any other use must be sought from the Advanced Placement Program[®]. Teachers may reproduce them, in whole or in part, in limited quantities for noncommercial, face-to-face teaching purposes. This permission does not apply to any third-party copyrights contained herein. This material may not be mass distributed, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here.

These materials were produced by Educational Testing Service® (ETS®), which develops and administers the examinations of the Advanced Placement Program for the College Board. The College Board and Educational Testing Service (ETS) are dedicated to the principle of equal opportunity, and their programs, services, and employment policies are guided by that principle.

The College Board is a national nonprofit membership association whose mission is to prepare, inspire, and connect students to college and opportunity. Founded in 1900, the association is composed of more than 4,300 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three million students and their parents, 22,000 high schools, and 3,500 colleges through major programs and services in college admissions, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT®, the PSAT/NMSQT®, and the Advanced Placement Program® (AP®). The College Board is committed to the principles of equity and excellence, and that commitment is embodied in all of its programs, services, activities, and concerns.

For further information, visit www.collegeboard.com

Copyright © 2003 College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Vertical Teams, APCD, Pacesetter, Pre-AP, SAT, Student Search Service, and the acorn logo are registered trademarks of the College Entrance Examination Board. AP Central is a trademark owned by the College Entrance Examination Board. PSAT/NMSQT is a registered trademark jointly owned by the College Entrance Examination Board and the National Merit Scholarship Corporation. Educational Testing Service and ETS are registered trademarks of Educational Testing Service. Other products and services may be trademarks of their respective owners.

For the College Board's online home for AP professionals, visit AP Central at apcentral collegeboard.com.

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

Complete function NumAlgaeAt below.

int Environment::NumAlgaeAt (const Position & pos) const

// precondition: pos is a valid position in the environment

// postcondition: returns the number of units of algae at pos

Ctum (my Algae E Pos. Row()) [105. (0)());

(b) Write the Fish member function MostAlgae, which is described as follows. MostAlgae should return a Position 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, fegardless of what you wrote in part (a). Complete function MostAlgae below.

```
Position Fish::MostAlgae(const Environment & env, const Neighborhood & nbrs) const

// precondition: nbrs.Size() > 0

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

fint [;

pos rax = nbrs. Select(0);

for (i=1; i L nbrs. Size(); i++)

if (env. Num Algae A+ (rax) < env. Num Algae A+ (nbrs. Select(i)))

Max = Nbrs. Select (i);

return max;
```

(c) Write the Fish member function Act, which is described as follows. If there is algae at the fish's current position, 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, Act will cause the fish to die by calling env. RemoveFish. If the fish does not eat and does not die, then the fish should move to a neighboring position with the most algae. Act should update the state of the environment and the state of the fish appropriately.

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

Complete function Act below.

```
void Fish::Act(Environment & env)
// precondition: this Fish is stored in env 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
11
                   consecutive step that this Fish did not eat,
                   then this Fish has been removed from env;
                   otherwise, this Fish moved.
//
                   myStepsSinceFed has been updated.
  if (env. Num Algae At (Location 11) > 0)
env. Remove Algae (Location 1);
       my Steps Since Fed = 0 ;
    else
        if (my Steps Since Fed == 2)
       env. Remove Fish (beation ());
else &
Move (env);
My Steps Since Fed ++;
```

B

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

Complete function NumAlgaeAt below.

```
int Environment::NumAlgaeAt (const Position & pos) const

// precondition: pos is a valid position in the environment

// postcondition: returns the number of units of algae at pos

{

return my Algae [ pos. Row() ][ pos. Col() ];

}
```

(b) Write the Fish member function MostAlgae, which is described as follows. MostAlgae should return a Position 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 function MostAlgae below.

```
Position Fish::MostAlgae(const Environment & env,
                          const Neighborhood & nbrs) const
// precondition: nbrs.Size() > 0
// postcondition: returns a Position from nbrs that contains
//
                  the most algae
  int algaecount = 0;
  Position pos;
  for (int i = 0; i < nors, Size(); i++)
      Position current = nbrs. select (i);
      if (env. Num Algae At (current) > algae count)
        algae count = env. Num Algae A + (current);
        pos = current;
      ξ.
   ξ
   return posj
```

Part (c) begins on page 22.

(c) Write the Fish member function Act, which is described as follows. If there is algae at the fish's current position, 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, Act will cause the fish to die by calling env. RemoveFish. If the fish does not eat and does not die, then the fish should move to a neighboring position with the most algae. Act should update the state of the environment and the state of the fish appropriately.

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

Complete function Act below.

```
void Fish::Act(Environment & env)
// precondition: this Fish is stored in env 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
11
                     consecutive step that this Fish did not eat,
11
11
                     then this Fish has been removed from env;
11
                     otherwise, this Fish moved.
                     myStepsSinceFed has been updated.
11
ş
    int algount = env. Num Algae At (Location (1);
    if (alg count > 0)

E env. Remove Algae (Location (), algcount - 1);
        my Steps Since Fed = 0;
env. Update (Location (), +this); // Update private var
     else if ( my steps since Fed == 3)
        env. Remove Fish (Location ());
     3
      else
        my Steps Since Fed + + ;
env. Update (Location (), # this );
       Move (env);
 ξ
```

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

Complete function NumAlgaeAt below.

```
int Environment::NumAlgaeAt(const Position & pos) const
// precondition: pos is a valid position in the environment
// postcondition: returns the number of units of algae at pos

{

return my Algae [ pos. Raw()][pos. Col()];
```

(b) Write the Fish member function MostAlgae, which is described as follows. MostAlgae should return a Position 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 function MostAlgae below.

```
Position Fish::MostAlgae(const Environment & env, const Neighborhood & nbrs) const

// precondition: nbrs.Size() > 0

// postcondition: returns a Position from nbrs that contains

// the most algae

Position P Most

Position P Most

Prosition P Most

Position P Most

Position P Most

Position From nbrs that contains

prosition from nbrs that contains

prosition from nbrs that contains

prosition from nbrs that contains

the most algae

Position From nbrs that contains

prosition from nbrs that contains

the most algae

Position P Most

prosition from nbrs that contains

prosition from nbrs that contains

the most algae

Position P Most

prosition from nbrs that contains

prosition from nbrs that contains

the most algae

Position P Most

prosition from nbrs that contains

pr
```

(c) Write the Fish member function Act, which is described as follows. If there is algae at the fish's current position, 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, Act will cause the fish to die by calling env. RemoveFish. If the fish does not eat and does not die, then the fish should move to a neighboring position with the most algae. Act should update the state of the environment and the state of the fish appropriately.

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

Complete function Act below.

```
void Fish::Act(Environment & env)
// precondition: this Fish is stored in env 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
11
                   consecutive step that this Fish did not eat,
                   then this Fish has been removed from env;
                   otherwise, this Fish moved.
                   myStepsSinceFed has been updated.
  if (my Steps Since Fed >= 3)
env. Remove Fish (Location ());
   else if (env. Num Algae A+ (Location()) > 0)

{
env. Remove Algae (Location(), 1);
       my Steps Since Fed = 0'
     ( Move (env);
my Steps Since Fed++;
```