



AP[®] Computer Science AB 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.

Complete function AllFishHelper below.

```
void Environment::AllFishHelper(Node * root, apvector<Fish> & fishList,
                                int & index) const
// precondition: 0 <= index < fishList.length();
//               there are no more than fishList.length() - index fish
//               in the subtree represented by root
// postcondition: All fish in the subtree represented by root have been
//               added to fishList in top-down, left-right order,
//               starting from index;
//               index has been increased by the number of fish
//               added to fishList
```

```
{
  if (root) AllFishHelper(root->left, fishList, index);
  AllFishHelper(root->right, fishList, index);
  fishList[index] = root->theFish;
  index++;
}
}
```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

Complete function AddFishHelper below.

```
void Environment::AddFishHelper(Node * & root, const Fish & fsh)
// precondition: root represents a subtree of a binary search tree;
//               nodes are ordered by fish position; no node in the
//               tree contains a fish at the same position as fsh
// postcondition: Fish fsh has been added to the subtree represented by
//               root, maintaining correct order of nodes
{
    if (!root)
    {
        root = new Node;
        root->theFish = fsh;
    }
    else if (fsh < root->theFish)
    {
        AddFishHelper(root->left, fsh);
    }
    else
    {
        AddFishHelper(root->right, fsh);
    }
}
```

GO ON TO THE NEXT PAGE.

Complete function AllFishHelper below.

```

void Environment::AllFishHelper(Node * root, apvector<Fish> & fishList,
                                int & index) const
// precondition: 0 <= index < fishList.length();
//               there are no more than fishList.length() - index fish
//               in the subtree represented by root
// postcondition: All fish in the subtree represented by root have been
//               added to fishList in top-down, left-right order,
//               starting from index;
//               index has been increased by the number of fish
//               added to fishList

{
    if (root == NULL) {
        return;
    }

    AllFishHelper (root -> left, fishList, index);
    fishList [index] = root -> theFish;
    index ++;
    AllFishHelper (root -> right, fishList, index);
}

```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

Complete function AddFishHelper below.

```
void Environment::AddFishHelper(Node * & root, const Fish & fsh)
// precondition: root represents a subtree of a binary search tree;
//               nodes are ordered by fish position; no node in the
//               tree contains a fish at the same position as fsh
// postcondition: Fish fsh has been added to the subtree represented by
//               root, maintaining correct order of nodes
```

```
{
```

```
    Node *temp = root;
    Node *parent = NULL;
    while (temp != NULL) {
        if (fsh < temp -> theFish) {
            parent = temp;
            temp = temp -> left;
        }
        else { // fsh is greater than temp -> theFish
            parent = temp;
            temp = temp -> right;
        }
    }
```

```
    if (parent == NULL) {
        root = new Node(fsh);
    }
```

```
    else {
        if (fsh < parent -> theFish) {
            parent -> left = new Node(fsh);
        }
        else {
            parent -> right = new Node(fsh);
        }
    }
```

```
}
```

end

GO ON TO THE NEXT PAGE.

Complete function AllFishHelper below.

```

void Environment::AllFishHelper(Node * root, apvector<Fish> & fishList,
                                int & index) const
// precondition: 0 <= index < fishList.length();
//               there are no more than fishList.length() - index fish
//               in the subtree represented by root
// postcondition: All fish in the subtree represented by root have been
//               added to fishList in top-down, left-right order,
//               starting from index;
//               index has been increased by the number of fish
//               added to fishList
{
    fishList [index] = root -> the Fish;
    index++;
    if (root -> left == NULL && root -> right == NULL)
        return;

    AllFishHelper(root -> left, fishList, index);
    AllFishHelper(root -> right, fishList, index);
}

```

Part (b) begins on page 18.

GO ON TO THE NEXT PAGE.

Complete function AddFishHelper below.

```
void Environment::AddFishHelper(Node * & root, const Fish & fsh)
// precondition: root represents a subtree of a binary search tree;
//               nodes are ordered by fish position; no node in the
//               tree contains a fish at the same position as fsh
// postcondition: Fish fsh has been added to the subtree represented by
//               root, maintaining correct order of nodes
{
    if (root == NULL)
    {
        Node * insert;
        insert -> theFish = fsh;
        root = insert;
        return;
    }
    else if (fsh < root -> theFish)
        AddFishHelper(root -> left, fsh)
    else
        AddFishHelper(root -> right, fsh)
}
```

GO ON TO THE NEXT PAGE.