



## AP Computer Science AB 1999 Sample Student Responses

**The materials included in these files are intended for non-commercial 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 face-to-face teaching purposes but may not mass distribute the materials, 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. This permission does not apply to any third-party copyrights contained herein.**

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 dedicated to preparing, inspiring, and connecting students to college and opportunity. Founded in 1900, the association is composed of more than 3,900 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 admission, guidance, assessment, financial aid, enrollment, and teaching and learning. Among its best-known programs are the SAT<sup>®</sup>, the PSAT/NMSQT<sup>™</sup>, the Advanced Placement Program<sup>®</sup> (AP<sup>®</sup>), and Pacesetter<sup>®</sup>. 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.

Copyright © 2001 by College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, and the acorn logo are registered trademarks of the College Entrance Examination Board.

- 6
- (a) Write function `PathLength`, as started below. If person `P` is in tree `T`, then `PathLength(T, P, 1)` should return the length of the longest path from the root of `T` to a node containing `P`; if person `P` does not appear in tree `T`, then `PathLength(T, P, 1)` should return 0. Note that parameter `level` can be used to keep track of the current level of the tree.

For the tree given above, the following are examples of calls to `PathLength`.

<u>Function Call</u>	<u>Value Returned</u>
<code>PathLength(T, "Susan", 1)</code>	4
<code>PathLength(T, "Ken", 1)</code>	3
<code>PathLength(T, "Chris", 1)</code>	6
<code>PathLength(T, "David", 1)</code>	0
<code>PathLength(T-&gt;left, "Theresa", 1)</code>	1
<code>PathLength(T-&gt;right-&gt;left, "Don", 1)</code>	3

In writing `PathLength`, you may call function `Max` as specified in the beginning of this question. Assume that `Max` works as specified.

Complete function `PathLength` below.

```
int PathLength(TreeNode * T, const apstring & someName, int level)
```

```
{
  int r;
  if (T == NULL)
    return 0;
  r = Max(PathLength(T->left, someName, level + 1), PathLength(T->right, someName, level + 1));
  if (r > 0) // got one
    return r;
  if (T->name == someName) // return n we
    return level;
  return 0; // nowhere below
}
```

Part (b) begins on page 20.

GO ON TO THE NEXT PAGE

- (b) Write function `RootPath`, as started below. `RootPath` should return the length of the longest path from the root of the tree to a node containing the same name as the root; if no node other than the root contains that name, then `RootPath` returns 1; if the tree is empty, `RootPath` should return 0. For the tree given above, the following are examples of calls to `RootPath`.

<u>Function Call</u>	<u>Value Returned</u>
<code>RootPath(T)</code>	4
<code>RootPath(T-&gt;left)</code>	1
<code>RootPath(T-&gt;right)</code>	5
<code>RootPath(T-&gt;left-&gt;left-&gt;left)</code>	0

In writing `RootPath`, you may call function `PathLength` specified in part (a). Assume that `PathLength` works as specified, regardless of what you wrote in part (a).

Complete function `RootPath` below.

```
int RootPath(TreeNode * T)
{
    if (T == NULL)
        return 0;
    else
        return PathLength(T, T->name, 1);
}
```

- (a) Write function `PathLength`, as started below. If person `P` is in tree `T`, then `PathLength(T, P, 1)` should return the length of the longest path from the root of `T` to a node containing `P`; if person `P` does not appear in tree `T`, then `PathLength(T, P, 1)` should return 0. Note that parameter `level` can be used to keep track of the current level of the tree.

For the tree given above, the following are examples of calls to `PathLength`.

<u>Function Call</u>	<u>Value Returned</u>
<code>PathLength(T, "Susan", 1)</code>	4
<code>PathLength(T, "Ken", 1)</code>	3
<code>PathLength(T, "Chris", 1)</code>	6
<code>PathLength(T, "David", 1)</code>	0
<code>PathLength(T-&gt;left, "Theresa", 1)</code>	1
<code>PathLength(T-&gt;right-&gt;left, "Don", 1)</code>	3

In writing `PathLength`, you may call function `Max` as specified in the beginning of this question. Assume that `Max` works as specified.

Complete function `PathLength` below.

```
int PathLength(TreeNode * T, const apstring & someName, int level)
```

```
if (T == NULL)
    return 0;
```

```
else
```

```
if (T->name == someName)
    return level;
```

```
else
```

```
{ level ++;
```

```
return max ( PathLength(T->left, someName, level),
             PathLength(T->right, someName, level) );
```

Part (b) begins on page 20.

GO ON TO THE NEXT PAGE

- (b) Write function `RootPath`, as started below. `RootPath` should return the length of the longest path from the root of the tree to a node containing the same name as the root; if no node other than the root contains that name, then `RootPath` returns 1; if the tree is empty, `RootPath` should return 0. For the tree given above, the following are examples of calls to `RootPath`.

<u>Function Call</u>	<u>Value Returned</u>
<code>RootPath(T)</code>	4
<code>RootPath(T-&gt;left)</code>	1
<code>RootPath(T-&gt;right)</code>	5
<code>RootPath(T-&gt;left-&gt;left-&gt;left)</code>	0

In writing `RootPath`, you may call function `PathLength` specified in part (a). Assume that `PathLength` works as specified, regardless of what you wrote in part (a).

Complete function `RootPath` below.

```
int RootPath(TreeNode * T)
```

```
{
    if (T == NULL)
        return 0;
    else
        return max(1, (PathLength(T, T->name, 1)));
}
```

- (a) Write function `PathLength`, as started below. If person `P` is in tree `T`, then `PathLength(T, P, 1)` should return the length of the longest path from the root of `T` to a node containing `P`; if person `P` does not appear in tree `T`, then `PathLength(T, P, 1)` should return 0. Note that parameter `level` can be used to keep track of the current level of the tree.

For the tree given above, the following are examples of calls to `PathLength`.

<u>Function Call</u>	<u>Value Returned</u>
<code>PathLength(T, "Susan", 1)</code>	4
<code>PathLength(T, "Ken", 1)</code>	3
<code>PathLength(T, "Chris", 1)</code>	6
<code>PathLength(T, "David", 1)</code>	0
<code>PathLength(T-&gt;left, "Theresa", 1)</code>	1
<code>PathLength(T-&gt;right-&gt;left, "Don", 1)</code>	3

In writing `PathLength`, you may call function `Max` as specified in the beginning of this question. Assume that `Max` works as specified.

Complete function `PathLength` below.

```
int PathLength(TreeNode * T, const apstring & someName, int level)
{
    if (T == NULL)
        return 0;

    PathLength(T->right, someName, level+1);
    PathLength(T->left, someName, level+1);
}
```

Part (b) begins on page 20.

GO ON TO THE NEXT PAGE

- (b) Write function `RootPath`, as started below. `RootPath` should return the length of the longest path from the root of the tree to a node containing the same name as the root; if no node other than the root contains that name, then `RootPath` returns 1; if the tree is empty, `RootPath` should return 0. For the tree given above, the following are examples of calls to `RootPath`.

<u>Function Call</u>	<u>Value Returned</u>
<code>RootPath(T)</code>	4
<code>RootPath(T-&gt;left)</code>	1
<code>RootPath(T-&gt;right)</code>	5
<code>RootPath(T-&gt;left-&gt;left-&gt;left)</code>	0

In writing `RootPath`, you may call function `PathLength` specified in part (a). Assume that `PathLength` works as specified, regardless of what you wrote in part (a).

Complete function `RootPath` below.

```
int RootPath(TreeNode * T)
```

```
{
```

```
    return PathLength(T, T->name, 1);
```

```
}
```