



AP[®] Computer Science AB 2004 Sample Student Responses

The materials included in these files are intended for noncommercial 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. This permission does not apply to any third-party copyrights contained herein. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here.

The College Board is a not-for-profit membership association whose mission is to connect students to college success and opportunity. Founded in 1900, the association is composed of more than 4,500 schools, colleges, universities, and other educational organizations. Each year, the College Board serves over three million students and their parents, 23,000 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 excellence and equity, and that commitment is embodied in all of its programs, services, activities, and concerns.

For further information, visit www.collegeboard.com

Copyright © 2004 College Entrance Examination Board. All rights reserved. College Board, Advanced Placement Program, AP, AP Central, AP Vertical Teams, APCD, Pacesetter, Pre-AP, SAT, Student Search Service, and the acorn logo are registered trademarks of the College Entrance Examination Board. PSAT/NMSQT is a registered trademark of the College Entrance Examination Board and 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 `VoterBallots` constructor. The constructor creates the `Map` `voteCount` and initializes it to hold one entry for each candidate that appears on at least one ballot in `ballotList`. The associated value for each candidate is the total number of ballots on which that candidate appears.

A solution that creates an unnecessary new instance of a `List` or `Set` will not receive full credit.

Complete the constructor below.

```
// precondition: each entry in ballotList is a Set representing
//               one voter's ballot
// postcondition: voteCount.get(candidate) is the total number of
//               times candidate appears on ballots in ballotList
public VoterBallots(List ballotList)
{
    Iterator list = ballotList.iterator();
    voteCount = new HashMap();
    while (list.hasNext())
    {
        Iterator itr = ((Set)list.next()).iterator();
        while (itr.hasNext())
        {
            String o = (String)itr.next();
            if (voteCount.containsKey(o))
            {
                voteCount.put(o, new Integer(((Integer)voteCount.get(o)).intValue() + 1));
            }
            else voteCount.put(o, new Integer(1));
        }
    }
}
```

Part (b) begins on page 12.

GO ON TO THE NEXT PAGE.

- (b) Write the `VoterBallots` method `candidatesWithMost`. Method `candidatesWithMost` returns a `Set` of the name(s) of the candidate(s) in `voteCount` with the most votes.

In writing method `candidatesWithMost`, you may call the private helper method `maxVotes`. Solutions that reimplement functionality provided by `maxVotes`, rather than invoking `maxVotes`, will not receive full credit. A solution that creates an unnecessary new instance of a `List` or `Set` will not receive full credit.

Complete method `candidatesWithMost` below.

```
// postcondition: returns a set containing the candidate(s)
//                  with the most votes
public Set candidatesWithMost()
{
    Integer max = maxVotes();
    Set most = new HashSet();
    Iterator itr = (voteCount.keySet()).iterator();
    while (itr.hasNext())
    {
        String o = (String)itr.next();
        if ((voteCount.get(o)).equals(max)) most.add(o);
    }
    return most;
}
```

GO ON TO THE NEXT PAGE.

A₃

- (c) Assume that there are C candidates and V voters. What is the expected time complexity, in Big-Oh notation in terms of C and V , for your implementation of method `candidatesWithMost`?

$$O(C)$$

GO ON TO THE NEXT PAGE.

AB 2 B.

- (a) Write the `VoterBallots` constructor. The constructor creates the `Map` `voteCount` and initializes it to hold one entry for each candidate that appears on at least one ballot in `ballotList`. The associated value for each candidate is the total number of ballots on which that candidate appears.

A solution that creates an unnecessary new instance of a `List` or `Set` will not receive full credit.

Complete the constructor below.

```
// precondition: each entry in ballotList is a Set representing
//               one voter's ballot
// postcondition: voteCount.get(candidate) is the total number of
//               times candidate appears on ballots in ballotList
public VoterBallots(List ballotList)
    Iterator itr = ballotList.iterator();
    while (itr.hasNext())
    {
        Iterator sit = itr.next().iterator();
        while (sit.hasNext())
        {
            String s = sit.next();
            if (voteCount.containsKey(s))
                voteCount.put(s, new Integer(++(voteCount.get(s).intValue())));
            else
                voteCount.put(s, new Integer(1));
        }
    }
}
```

Part (b) begins on page 12.

GO ON TO THE NEXT PAGE.

- (b) Write the VoterBallots method candidatesWithMost. Method candidatesWithMost returns a Set of the name(s) of the candidate(s) in voteCount with the most votes.

In writing method candidatesWithMost, you may call the private helper method maxVotes. Solutions that reimplement functionality provided by maxVotes, rather than invoking maxVotes, will not receive full credit. A solution that creates an unnecessary new instance of a List or Set will not receive full credit.

Complete method candidatesWithMost below.

```
// postcondition: returns a set containing the candidate(s)
//                  with the most votes
public Set candidatesWithMost()
    Iterator itr = voteCount.iterator();
    Integer m = maxVotes();
    Iterator sit = voteCount.keySet().iterator();
    while (sit.hasNext()) {
        if (!voteCount.get(sit.next()).equals(m))
            sit.remove();
    }
    return voteCount.keySet();
```

GO ON TO THE NEXT PAGE.

- (c) Assume that there are C candidates and V voters. What is the expected time complexity, in Big-Oh notation in terms of C and V , for your implementation of method `candidatesWithMost`?

$O(C)$

GO ON TO THE NEXT PAGE.

- (a) Write the VoterBallots constructor. The constructor creates the Map voteCount and initializes it to hold one entry for each candidate that appears on at least one ballot in ballotList. The associated value for each candidate is the total number of ballots on which that candidate appears.

A solution that creates an unnecessary new instance of a List or Set will not receive full credit.

Complete the constructor below.

```
// precondition: each entry in ballotList is a Set representing
//               one voter's ballot
// postcondition: voteCount.get(candidate) is the total number of
//               times candidate appears on ballots in ballotList
public VoterBallots(List ballotList)
```

```
{ Map temp;
  for (int x=0; x < list.size(); x++)
  {
    Iterator iter = new Iterator(list.get(x));
    while (iter.hasNext())
    {
      String name = iter.next();
      if (temp.containsKey(name))
        temp.put(name, 1);
      else {
        int trash = temp.get(name) + 1;
        temp.put(name, trash);
      }
    }
    voteCount = temp;
  }
}
```

Part (b) begins on page 12.

GO ON TO THE NEXT PAGE.

C₂

- (b) Write the VoterBallots method candidatesWithMost. Method candidatesWithMost returns a Set of the name(s) of the candidate(s) in voteCount with the most votes.

In writing method candidatesWithMost, you may call the private helper method maxVotes. Solutions that reimplement functionality provided by maxVotes, rather than invoking maxVotes, will not receive full credit. A solution that creates an unnecessary new instance of a List or Set will not receive full credit.

Complete method candidatesWithMost below.

```
// postcondition: returns a set containing the candidate(s)
//                  with the most votes
public Set candidatesWithMost()
```

```
{
```

```
    Set temp;
```

```
    for (int x=0; x < voteCounts.size(); x++)
```

```
    {
        if (voteCounts.maxVotes == voteCounts.get(x),
```

```
            temp.add(voteCounts.get(x));
```

```
    }
```

```
}
```

GO ON TO THE NEXT PAGE.

C₃

- (c) Assume that there are C candidates and V voters. What is the expected time complexity, in Big-Oh notation in terms of C and V , for your implementation of method `candidatesWithMost`?

$$O(\log(V) C^2)$$

GO ON TO THE NEXT PAGE.