AP® Computer Science AB
2004 Sample Student Responses

AB 2 A,

(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.

A2

(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)

AB 2 B1

(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(0));
       }
    }
```

Part (b) begins on page 12.

(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();
```

B3

(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 C1

(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 = (int) temp.get (name) + 1;
              temp.put (name, trash)
         }
    }
voteCount = temp;
}
```

Part (b) begins on page 12.

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`, wil 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));
          }
   }
```

C_3

(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`?

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