

## Sample Free-Response Questions

Following is a representative set of questions. Questions marked with an asterisk are also representative of AB exam questions. The AP Computer Science A Exam will include one free-response question based on the *AP Computer Science Case Study*. (See AP Central for examples.)

*Directions:* SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

### Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
  - Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
  - In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.
1. In an instant runoff election there are two or more candidates and there are many voters. Each voter votes by submitting a ballot that is an ordered list of all the candidates, where the first name listed is the voter's first choice, the second name is the voter's second choice, and so on. There are no ties allowed on a voter's ballot.

The election is decided by the following process.

- Initially, all candidates are placed on the current candidate list.
- As long as there are two or more candidates on the current candidate list, the following steps are repeated.
  1. Each ballot is examined for candidates on the current candidate list and a vote is counted for the current candidate that appears earliest in the list of names on the ballot. (On the first pass, this will be the first name on the ballot. In subsequent passes, it might not be the first name on the ballot. See the illustrations below.)
  2. The candidate(s) with the fewest votes is (are) eliminated from the current candidate list.
- The last remaining candidate is the winner. If there is none, the election is not decisive.

For example, suppose there are four candidates in the election: Chris, Jamie, Pat, and Sandy. Each ballot has these four names listed in order of the voter's preference, with the first choice appearing first in the list. Assume that seven ballots were submitted as shown in the following table.

**Current Candidate List:** Chris, Jamie, Pat, Sandy

<b>Voter</b>	<b>Ballot</b>	<b>First Choice from Current Candidate List</b>
0	Chris, Jamie, Pat, Sandy	Chris
1	Chris, Pat, Sandy, Jamie	Chris
2	Chris, Sandy, Pat, Jamie	Chris
3	Pat, Jamie, Sandy, Chris	Pat
4	Pat, Sandy, Chris, Jamie	Pat
5	Sandy, Pat, Jamie, Chris	Sandy
6	Jamie, Sandy, Pat, Chris	Jamie

In the first pass, Chris has 3 votes, Pat has 2 votes, Sandy has 1 vote, and Jamie has 1 vote. Jamie and Sandy are tied for the fewest votes; so both are eliminated, leaving Chris and Pat on the current candidate list. Voter preferences for these two candidates are shown in the following table.

**Current Candidate List:** Chris, Pat

<b>Voter</b>	<b>Ballot</b>	<b>First Choice from Current Candidate List</b>
0	Chris, Jamie, Pat, Sandy	Chris
1	Chris, Pat, Sandy, Jamie	Chris
2	Chris, Sandy, Pat, Jamie	Chris
3	Pat, Jamie, Sandy, Chris	Pat
4	Pat, Sandy, Chris, Jamie	Pat
5	Sandy, Pat, Jamie, Chris	Pat
6	Jamie, Sandy, Pat, Chris	Pat

In the second pass, Chris has 3 votes and Pat has 4 votes. Chris has fewest votes and is eliminated. Pat is the only remaining candidate and is therefore the winner of the election.

A ballot is modeled with the following partial class declaration.

```
public class Ballot
{
    /** @param candidateList a list of candidate names
     *   @return the name of the first choice candidate for this Ballot
     *           from those in candidateList
     */
    public String firstChoiceFrom(ArrayList<String> candidateList)
    { /* implementation not shown */ }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

The Ballot method `firstChoiceFrom` returns the name of the candidate from `candidateList` that appears first on this ballot.

The set of ballots for all voters in an election is modeled with the following partial class declaration.

```
public class VoterBallots
{
    private ArrayList<Ballot> ballotList;
    // each entry represents one voter's ballot

    /** @param candidate the name of a candidate
     *   @param candidateList a list of candidate names
     *   Precondition: candidate appears in candidateList
     *   @return the number of times that candidate is first among
     *           those in candidateList for elements of ballotList
     */
    private int numFirstVotes(String candidate,
                              ArrayList<String> candidateList)
    { /* to be implemented in part (a) */ }

    /** @param candidateList a list of candidate names
     *   Precondition: each String in candidateList appears exactly
     *                   once in each Ballot in ballotList
     *   @return a list of those candidates tied with the fewest first choice votes
     */
    public ArrayList<String> candidatesWithFewest(
        ArrayList<String> candidateList)
    { /* to be implemented in part (b) */ }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

An instant runoff election is represented by the class `InstantRunoff` that encapsulates the process of selecting a winner by repeatedly applying the `VoterBallots` method `candidatesWithFewest` to a list of candidates that is reduced until only the winner remains. This class is not shown here.

- (a) Write the `VoterBallots` method `numFirstVotes`. Method `numFirstVotes` should return the number of times `candidate` appears first, among those elements that are on `candidateList`, in elements of `ballotList`.

Complete method `numFirstVotes` below.

```
/** @param candidate the name of a candidate
 * @param candidateList a list of candidate names
 * Precondition: candidate appears in candidateList
 * @return the number of times that candidate is first among
 * those in candidateList for elements of ballotList
 */
private int numFirstVotes(String candidate,
                          ArrayList<String> candidateList)
```

- (b) Write the `VoterBallots` method `candidatesWithFewest`. Method `candidatesWithFewest` should count the number of times each `String` in the list `candidateList` appears first in an element of `ballotList`, and return an `ArrayList` of all those `Strings` that are tied for the smallest count.

In writing method `candidatesWithFewest` you may use the private helper method `numFirstVotes` specified in part (a). Assume that `numFirstVotes` works as specified, regardless of what you wrote in part (a).

Complete method `candidatesWithFewest` below.

```
/** @param candidateList a list of candidate names
 * Precondition: each String in candidateList appears exactly
 * once in each Ballot in ballotList
 * @return a list of those candidates tied with the fewest first choice votes
 */
public ArrayList<String> candidatesWithFewest (
    ArrayList<String> candidateList)
```