Name _____

Date _____ Block _____

Directions: For each question: be clear, syntactically correct and direct. This is a 50 minute exam. Marshal your time!

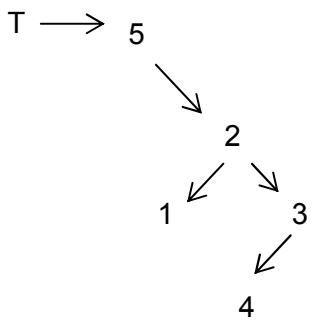
1. This question assumes that binary trees are implemented using the `TreeNode` class defined by the College Board.

a) Write method `TreeHeight` that returns the height of a tree where `TreeHeight` is defined as the number of levels contained in a tree. For example, an empty tree's `TreeHeight` is 0 and a tree with only one node has a `TreeHeight` of 1. The two trees pictured in section b of this question would each have a `TreeHeight` of 4.

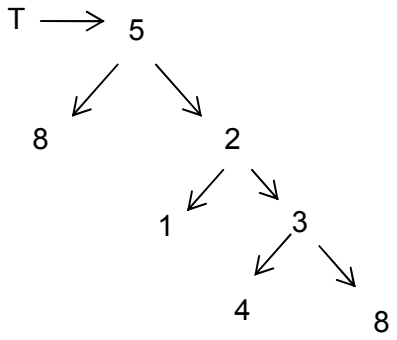
```
public int TreeHeight( TreeNode T )
// pre:  T is the root of a binary tree
// post: the height of T is returned
{
```

b) Write a method `AddSibling` that takes a binary tree and adds to all nodes with only one child another child containing the given value. For example,

Starting Tree T

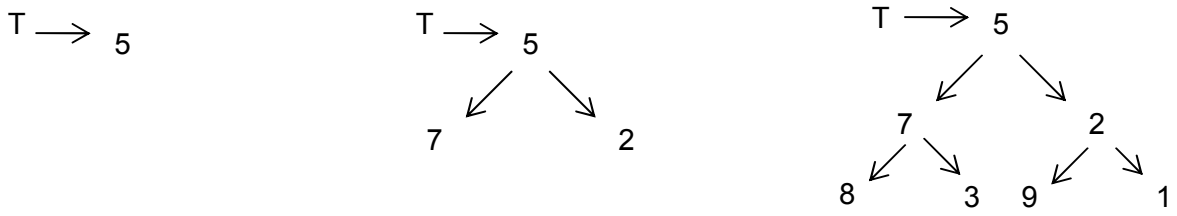


After a call to `AddSibling(T, 8)`




```
public void AddSibling( TreeNode T, int Value )
// pre:  T is empty or points to the root of a binary
// post: Another child containing Value has been added to each node in T
//       that previously had exactly one child
```

c) A binary tree is said to be full if all its levels are filled with nodes. A full tree contains the maximum possible number of nodes for a binary tree of a certain height. For example, these are all full trees:



Write method `IsFullTree` as stated below. In writing method `IsFullTree`, you may assume that `TreeHeight` works as specified, no matter what you wrote in part a.

```
public boolean IsFullTree( TreeNode T )
// pre:  T points to a non-empty binary tree
// post: returns true if T is a full tree, false otherwise
{
```

2. 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

Voter	**Ballot**	**First Choice from Current Candidate List**
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

Voter	**Ballot**	**First Choice from Current Candidate List**
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
// postcondition: returns the first choice candidate for this Ballotcfrom
// those on the candidateList
{
  public String firstChoiceFrom( ArrayList candidateList ) { /* code not shown */ }
  // ... constructors, other methods,
  //    and private data 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 ballotList;
  // each entry is an instance of Ballot representing one voter's ballot

  private int numFirstVotes(String candidate, ArrayList candidateList)
  // precondition: candidate appears in candidateList
  // postcondition: returns the number of times that candidate is first
  //  among those on candidateList for elements of ballotList
  { /* to be implemented in part (a) */ }


  public ArrayList candidatesWithFewest( ArrayList candidateList )
  // precondition: each String in candidateList appears exactly once in
  // each Ballot in ballotList
  // postcondition: returns a list of those candidates tied with the fewest
  // first choice votes
  {/* to be implemented in part (b) */ }

  // ... constructor(s) and other methods 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.

```
private int numFirstVotes(String candidate,  ArrayList candidateList)
// precondition: candidate appears in candidateList
// postcondition: returns the number of times that candidate is first among those
// on candidateList for elements of ballotList
{
```

(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). Solutions that reimplement functionality provided by this method, rather than invoking it, will not receive full credit.

Complete method `candidatesWithFewest` below.

```
public ArrayList candidatesWithFewest( ArrayList candidateList)
// precondition: each String in candidateList appears exactly once in each Ballot
// in ballotList
// postcondition: returns a list of those candidates tied with the fewest first
// choice votes
{
```