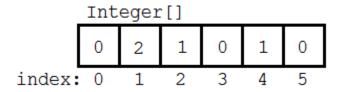
## 5 Vote Iterator (10 points)

Write an iterator that takes in an Integer array of vote counts and iterates over the votes. The input array contains the number of votes each selection received. For example, if the input array contained the following:



then calls to next() would eventually return 1 twice (because at index 1, the input array has value 2), 2 once, and 4 once. After that, hasNext() would return false.

Provide code for the VoteIterator class below. Make sure your iterator adheres to standard iterator rules.

## **Solution:**

```
import java.util.ArrayList;
import java.util.Iterator;
public class VoteIterator implements Iterator<Integer> {
    private Integer[] inputVotes;
    private ArrayList<Integer> votesArrayList;
    private int votesIndex;
    public VoteIterator(Integer[] input) {
        votesArrayList = new ArrayList<Integer>();
        for (int i = 0; i < input.length; i++) {</pre>
            for (int j = 0; j < input[i]; j++) {
                votesArrayList.add(new Integer(i));
            }
        votesIndex = 0;
        inputVotes = input;
    }
    public boolean hasNext() {
        return votesIndex < votesArrayList.size();</pre>
    }
```

```
public Integer next() {
    if (hasNext()) {
        votesIndex++;
        return votesArrayList.get(votesIndex - 1);
    } else {
        return null; // or throw some exception
    }
}

public void remove() {
    if (votesIndex > 0) {
        inputVotes[votesArrayList.get(votesIndex - 1)]--;
    }
}
```

**Comments:** This solution is the simplest in terms of code. It does a lot of preprocessing in the constructor by adding all of the votes to an ArrayList and then returns the votes one at a time with calls to next. Note that you still have to store the original input array for remove to work properly.

## **Alternate solution:**

```
import java.util.Iterator;
public class VoteIterator implements Iterator<Integer> {
    private Integer[] votes;
    private int voteIndex; // Which vote within same vote type
    private int bucketIndex; // Type of vote
    public VoteIterator(Integer[] input) {
        votes = input;
        voteIndex = 0;
        bucketIndex = 0;
    }
    public boolean hasNext() {
        if (voteIndex < votes[bucketIndex]) {</pre>
            return true;
        } else {
            for (int i = bucketIndex + 1; i < votes.length; i++) {</pre>
                if (votes[i] > 0) {
                    return true;
            }
        return false;
```

```
}
    public Integer next() {
        if (voteIndex < votes[bucketIndex]) {</pre>
            voteIndex++;
            return bucketIndex;
        } else {
            for (int i = bucketIndex + 1; i < votes.length; i++) {</pre>
                if (votes[i] > 0) {
                    bucketIndex = i;
                    voteIndex = 1;
                    return bucketIndex;
        }
        return null; // or throw some exception
    }
   public void remove() {
        if (voteIndex == 0) {
            return; // or throw some exception
        } else {
            votes[bucketIndex]--;
            voteIndex--;
        }
    }
}
```

**Comments:** This is a more conceptually straightforward solution that iterates through the input votes array. It keeps two counters: one that indexes into the input votes array and another that keeps track of how many votes of the current vote type have already been returned.