PROBLEM 3:  *(The Long and Short of it (10 points))*

In this problem assume that a method `convertAll` is implemented. It returns an array of all the proteins in a dna-sequence (you may have written a similar method for extra credit, and we discussed a similar method in class.) The returned array may have zero proteins or 100 proteins. In the version of `convertAll` below, an ORF is specified.

```java
/**
 * Returns all proteins found in dna for specified orf
 * @param dna is the sequence from which proteins are obtained
 * @param orf is the open reading frame used in finding proteins
 */
public String[] convertAll(String dna, int orf){
    // implementation not shown, but convertAll can be called
}
```

Part A (6 points)

One part of determining the best ORF for a protein is the number of proteins coded using the ORF. Write a method `bestORF` which returns which of the ORFs 1, 2, or 3 is the best for a particular DNA sequence. The best ORF is the one for which the most proteins are found. You may call `convertAll` shown above, assume it works as specified.

```java
/**
 * Returns the best ORF for dna, where best decided by which ORF codes the most proteins
 * (considering only ORFs 1,2,3)
 * @param dna for which the best ORF is found
 * @return the best ORF
 */
public int bestORF(String dna){
}
```
Part B (4 points)
An alternative definition of best ORF is based on which ORF produces a group of proteins with the maximal median length. So, an ORF which codes proteins whose lengths are 20, 20, 50, 60, 70 (median length is 50) is better than an ORF which codes proteins whose lengths are 10, 10, 15, 15, 20, 20, 30, 30, 50, 50, 70 (median length is 20). The median is the middle value of a sorted list.
Don’t write code, but describe the code/method you’d write to determine the best ORF using these criteria. Be precise, but don’t write code.
In this problem you’ll need to use the class `ProteinTool` shown below by calling the static methods appropriately in solving problems related to protein hydrophilicity – the extent to which proteins have an affinity for water.

```java
public class ProteinTool {

    public static double hydrophilicity(String p) {
        // implementation not shown
    }

    public static double hoppWoodValue(String p) {
        // implementation not shown
    }
}
```

no code to write on this page
Part A (6 points)
Write the method `hydroCount` which calculates the number of proteins whose hydrophilicity is above a specified threshold and returns this number. The array parameter `plist` is an array of proteins, parameter `thresh` is the minimal value for which a protein should be considered hydrophilic. For example, if `plist` stores 10 proteins whose hydrophilicity indexes (as obtained from `ProteinTool.hydrophilicity`) are as follows:

| 2.0 | 1.5 | 2.2 | 1.7 | 2.3 | -1.5 | -3.0 | -2.0 | 1.5 | 1.8 |

then the call `hydroCount(plist,-2.5)` should return 9, the call `hydroCount(plist,2.1)` should return 2 and the call `hydroCount(plist,2.8)` should return 0. In writing `hydroCount` you’ll need to call `ProteinTool.hydrophilicity` to determine a protein’s hydrophilicity.

```java
/**
 * Return number of proteins in plist with hydrophilicity above thresh
 * @param plist is an array of proteins
 * @param thresh is the hydrophilicity threshold
 */
public int hydroCount(String[] plist, double thresh){

```
```