Problem 1, Part D:

```
static void Problem1Sort(int[] a, int arraySize){
    int temp = 0;
    String strArray[] = new String[arraySize];
    for (int i = 0; i < arraySize; i++) {
        strArray[i] = String.valueOf(a[i]);
    }
    int elementLength = strArray[0].length();
    for (int j = 1; j < strArray.length; j++){
        if (strArray[j].length() > elementLength){
            elementLength = strArray[j].length();
        }
    }
    int k = 10;
    while (k <= Math.pov(10,elementLength)) {
        for (int i = 0; i < arraySize; i++) {
            for (int j = i + 1; j < arraySize; j++) {
                int x = a[i] % k;
                int y = a[j] % k;
                if (x > y) {
                    temp = a[i];
                    a[i] = a[j];
                    a[j] = temp;
                    System.out.println(Arrays.toString(a));
                }
            }
        }
```

There exists two stand alone “for” loops, which means that the complexity of the two of those equates out to $n^2 + n^2 = 2n^2$

Beneath that, there are two “for” loops nested into each other—both of which are nested in the “while” loop, which gives a complexity of $3n^2$. The total complexity is $O(n^2)$

It is stable since the loops shown above works for all values.

Problem 1, Part E:

The program itself uses a constant space in the beginning, If it used $O(n)$ space the runtime would be longer.