

1. Review the Java code below, identify any errors, circle them, and explain why they are incorrect.

This is an example

```
void HandleStuff( CORP_DATA & inputRec, int crntQtr, EMP_DATA empRec, double
& estimRevenue, double ytdRevenue, int screenX, int screenY, COLOR_TYPE &
newColor, COLOR_TYPE & prevColor, StatusType & status, int expenseType )
{
    int i;
    for ( i = 0; i < 100; i++ ) {
        inputRec.revenue[i] = 0;
        inputRec.expense[i] = corpExpense[ crntQtr ][ i ];
    }
    UpdateCorpDatabase( empRec );
    estimRevenue = ytdRevenue * 4.0 / (double) crntQtr;
    newColor = prevColor;
    status = SUCCESS;
    if ( expenseType == 1 ) {
        for ( i = 0; i < 12; i++ )
            profit[i] = revenue[i] - expense.type1[i];
    }
    else if ( expenseType == 2 ) {
        profit[i] = revenue[i] - expense.type2[i];
    }
    else if ( expenseType == 3 )
        profit[i] = revenue[i] - expense.type3[i];
    }
```

7. Only CORP_DATA is used!!

1. Bad Name!!

5. Doesn't defend itself against BAD data!! (crntQtr = 0! Error!)

2. Input variable is changed!

3. Read&Write Global Variable (corpExpense, profit)

4. Not Single purpose - Write to DB, Calculate Xxx

6. Magic Number!! 100, 4.0, 12, 123?

This is the code that you must review

```
1 public class ArrayExamples
2 { public static void main(String[] args)
3 {   int[] list = {1, 2, 3, 4, 1, 2, 3};
4     findAndPrintPairs(list, 5);
5     bubblesort(list);
6     showList(list);
7
8     list = new int[]{ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};
9     bubblesort(list);
10    showList(list);
11
12    list = new int[]{11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2};
13    bubblesort(list);
14    showList(list);
15
16    list = new int[]{1};
17    bubblesort(list);
18    showList(list);
19 }
20
21 public static int findMin(int[] list)
22 {   assert list != null && list.length > 0 : "failed precondition";
23     int indexofmin = 0;
24     for(int i = 1; i < list.length; i++)
25     {   if(list[i] < list[indexofmin])
26         {   indexofmin = i;
27         }
28     }
29     return indexofmin;
30 }
31
32 public static void BadResize(int[] list, int newSize)
33 {   assert list != null && newSize >= 0 : "failed precondition";
34     int[] temp = new int[newSize];
35     int limit = Math.min(list.length, newSize);
36
37     for(int i = 0; i < limit; i++)
38     {   temp[i] = list[i];
39     }
40     list = temp;
41 }
42
43 public static int[] GoodResize(int[] list, int newSize)
44 {   assert list != null && newSize >= 0 : "failed precondition";
45     int[] result = new int[newSize];
46     int limit = Math.min(list.length, newSize);
47
48     for(int i = 0; i < limit; i++)
49     {   result[i] = list[i];
50     }
51
52     return result;
53 }
54
55 public static void findAndPrintPairs(int[] list, int target)
56 {   assert list != null : "failed precondition";
57     for(int i = 0; i < list.length; i++)
58     {   for(int j = i + 1; j < list.length; j++)
59         {   if(list[i] + list[j] == target)
60             {   System.out.println("The two elements at indices " + i + " and "
61                 + j
62                 + " are " + list[i] + " and " + list[j] + " add up to " +
63                 target);
64             }
65         }
66     }
67 }
```

Method names should follow Java naming conventions (camelCase)

Bad naming conventions

should be int[]

should be return temp directly

Method names should follow Java naming conventions (camelCase)

```
66
67
68 public static void bubblesort(int[] list)
69 {   assert list != null : "failed precondition";
70     int temp;
71     boolean changed = true;
72     for(int i = 0; i < list.length && changed; i++)
73     {   changed = false;
74         for(int j = 0; j < list.length - i - 1; j++)
75         {   assert (j > 0) && (j + 1 < list.length) : "loop counter j " + j +
76             "is out of bounds.";
77             if(list[j] > list[j+1])
78             {   changed = true;
79                 temp = list[j + 1];
80                 list[j + 1] = list[j];
81                 list[j] = temp;
82             }
83         }
84     }
85     assert isAscending( list );
86 }
87
88 public static void showList(int[] list)
89 {   for(int i = 0; i < list.length; i++)
90     System.out.print( list[i] + " " );
91     System.out.println();
92 }
93
94 public static boolean isAscending( int[] list )
95 {   boolean ascending = true;
96     int index = 1;
97     while( ascending && index < list.length )
98     {   assert index >= 0 && index < list.length;
99         ascending = (list[index - 1] <= list[index]);
100         index++;
101     }
102     return ascending;
103 }
104 }
```