

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

Method names should follow Java naming conventions (camelCase)

Bad naming conventions

should be return temp directly

should be int[]

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 }
```