Solutions - Problem Set 4

Total: 40 points

Problem R7.6 (R7.2) -- Array Loops: 4 points

a) 25

b) 13

c) 12

d) gives ArrayIndexOutOfBoundsException; theoretically total remains 0

e) 11

f) 25

g) 12

h) -1

Problem R7.13 (7.9) -- Enhanced for: 4 points

a)

```
for (float elem : values) {
     total = total + elem;
}
```

b)

```
boolean first = true;
for (float elem : values) {
     if (first == false) {
          total = total + elem;
     }
     else {
          first = false;
     }
}
```

c)

```
int index i = 0
for (float elem : values) {
     if (elem == target) {
          return i;
     }
     i++;
}
```

Problem R7.23 (R7.22) -- Longest Run: 4 points

(example)

```
[1, 1, 2, 2, 2, 3, 3, 3] -> 3 is length of the longest run

[1, 1, 1, 1, 2, 2] -> 4 is the length of the longest run
```

(pseudocode)

```
int[] myArr
maxRun = 1
currRun = 1
previous = myArr[0]
for (i = 1 to myArr.length):
      currRun = 1
      if (myArr[i] == previous):
            currRun++
      else:
            currRun = 1
      previous = myArr[i]
      if currRun > maxRun:
            maxRun = currRun
```

Problem R7.32 (R7.31) True/False: 4 points

Format: (answer) -- (question)

a) **True** -- All elements of an array are of the same type
b) **False** -- Arrays cannot contain strings as elements.
c) **False** -- Two-dimensional arrays always have the same number of rows and columns.
d) **False** -- Elements of different columns in a two-dimensional array can have different types.
e) **False** -- A method cannot return a two-dimensional array.
f) **True** -- A method cannot change the length of an array argument.
g) **True** -- A method cannot change the number of columns of an argument that is a two-dimensional array.

Problem R7.33 (R7.32) -- ArrayList Methods: 4 points

a)

```
ArrayList<Integer> firstArrayList = new
ArrayList<>(Arrays.asList(1,2,3,4));
     ArrayList<Integer> secondArrayList = new
ArrayList<>(Arrays.asList(1,2,3,4));

if (firstArrayList.size() != secondArrayList.size())
{
     return false;
}
for(int i = 0; i < firstArrayList.size(); i++)
{
     if (firstArrayList.get(i) != secondArrayList.get(i))
     {
          return false;
     }
}
```

b)

```
     // deep copy
     ArrayList<Integer> firstArrayList = new
ArrayList<>(Arrays.asList(1,2,3,4));
     ArrayList<Integer> secondArrayList = new ArrayList<>();
          for (int elem : firstArrayList) {
               secondArrayList.add(elem);
          }
```

c)

```
     ArrayList<Integer> firstArrayList = new
ArrayList<>(Arrays.asList(1,2,3,4));
     for (int i = 0; i < firstArrayList.size(); i++) {
          firstArrayList.set(i, 0);
     }
```

d)

```
     ArrayList<Integer> firstArrayList = new
ArrayList<>(Arrays.asList(1,2,3,4));
     firstArrayList.clear();

// can also do loop like this
While (firstArrayList.size() != 0)
{
     firstArraylist.remove(0)
}
```