COMP 248 - Tutorial #8 - Solution

Arrays

Question 1: What is the output of the following code?

```
class Secret
{
  public static void main(String args[])
  {
    int[] n = {4, 2, 6, 12, 0, -4, 6, 3, 8, 1};

    System.out.println("Array before:");
    for (int i = 0; i < n.length; i++) {
      System.out.println("n["+i+"] = " + n[i]);
    }

    boolean secretProperty = false;

    while (!secretProperty) {
      secretProperty = true;
      for (int i=0; i < (n.length-1); i++) {
        if (n[i] > n[i+1]) {
          int temp = n[i];
          n[i] = n[i+1];
          n[i+1] = temp;
          secretProperty = false;
        }
      }
    }

    System.out.println();
    System.out.println("Array after:");
    for (int i = 0; i < n.length; i++) {
       System.out.println("n["+i+"] = " + n[i]);
    }
  }
}
```

Answer: This will actually sort the array in ascending order. This algorithm is actually called Bubble sort.

Question 2: What will be displayed by the following segment of code?

```
int i;
int a[] = {5, 2, 3, 1, 1, 0, 2, 1, 0, 1};
for (i = 0; (i < a.length); i++)
{
      if (a[i] == 0)
         break;
      if (i % 3 == 0)
         continue;
      System.out.print(a[i]);
}
```

Answer: 231

Question 3: What will be displayed by the following ?

```
int[] data = {1,3,5,8,11,15};
int sum = 0;
for (int i = 1; i < data.length; ++i) {
   sum = sum + data[i] - data[i-1];
   System.out.println("sum    = " + sum);
}
```

Solution:

```
sum  = 2
sum  = 4
sum  = 7
sum  = 10
sum  = 14
```

Question 4: Consider the following fragment of Java code:

```
 int [] x = {0,0,1,1,1,  1,1,1,1,1,  1,1,1,1,1,  1,1,1,1,1,
              1,1,1,1,1,   1,1,1,1,1,  1,1,1,1,1,  1,1,1,1,1,
              1,1,1,1,1, 1,1,1,1,1}; // x has 50 elements
  int i, t;

  for (i=2; i<8; i++)                 // line 1
    if (x[i]!=0){                     // line 2
      System.out.print(i+ " ");       // line 3 – for question A
      t=2*i;                          // line 4
      while (t<=100){                 // line 5
        x[t]=0;                       // line 6
        t+=i;                         // line 7
      }
    }
  System.out.println();               // line 8 – for question B

  for (i=2; i<=50; i++)
    if (x[i]!=0)
      System.out.println(i);
```

A What is the output after the execution of the first `System.out.println` statement (on line 3) ?

Solution:

2 (because the first element in the array has an index 0)

B How would you describe the list of numbers output by the second `System.out.println` statement (on line 8) ?

Solution:

```
2
3
4
5
6
7
8
9
10
11
12
...
47
48
49
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 50
        at Parray3.main(Ass3.java:20)
```

(The program craches after trying to access the element at the index 50 because the last element of this array has index 49).

Question 5: What is the output of the following code?

```
public class ArrayOfArraysAnimalDemo
{
   public static void main(String[] args)
   {
      String[][] animals = {
                  { "dog", "cat", "fish", "bird", "worm" },
                  { "lion", "baboon", "bison", "beaver" },
                  { "bear", "bat", "ant", "bobcat", "buffalo",
"elephant"},
                  { "crab", "coyote", "cow", "frog", "goat",
"grissly" }
               };


      for (int i = 0; i < animals.length; i++)
      {
         System.out.print("["+i+"]["+0+"]" + animals[i][0] + " --
");
         for (int j = 1; j < animals[i].length; j++)
         {
            System.out.print("["+i+"]["+j+"]" + animals[i][j] + "
");
         }
         System.out.println();
      }
   }
}
```

Answer:

```
[0][0]dog -- [0][1]cat [0][2]fish [0][3]bird [0][4]worm
[1][0]lion -- [1][1]baboon [1][2]bison [1][3]beaver
[2][0]bear -- [2][1]bat [2][2]ant [2][3]bobcat [2][4]buffalo
[2][5]elephant
[3][0]crab -- [3][1]coyote [3][2]cow [3][3]frog [3][4]goat
[3][5]grissly
```

Question 6: Write the necessary statement to perform the following operations on single-dimension arrays:

A- Declare and initialise an array of 10 integers with the values -10.

```
int[] theArray = new int[10];
for (int i = 0; i < theArray.length; i++)
theArray[i] = -10;
```

B- Add 1 to each of the 20 elements of an integer array called **`values`**.

```
for (int i = 0; i < values.length; i++)
values[i]++;
```

C- Read 7 values for a float array called **`dailyTemperatures`** from the keyboard.

```
for (int i = 0; i < dailyTemperatures.length; i++)
dailyTemperatures[i] = myKeyboard.nextFloat();
```

D- Print the 5 values of an integer array called **`bestScores`** in column format.

```
for (int i = 0; i < bestScores.length; i++)
  System.out.println(bestScores[i]);
```

Question 7: Write a program to reverse the elements of an integer array. Note, your program should not just display the elements in reverse order but actually change the content of the array. For example, if the array contains:

1	2	3	4

then after your program, the array should contain:

4	3	2	1

```
int temp;
for (int i=0; i<theArray.length/2; i++){
    temp =  theArray[i];
    theArray[i] = theArray[theArray.length-1-i];
     theArray[theArray.length-1-i] = temp;
}
```

Question 8: Write a program to add the elements on the two diagonals of a square two dimensional integer array and display that sum. Ensure that if there is a middle element in the array it is not counted twice in the sum.

For example, with the array:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

your program should display 68 (1+6+11+16+4+7+10+13).

With the array:

1	2	3
5	6	7
9	10	11

your program should display 30 (1+6+11+3+9) (note that the 6 is not counted twice).

```
int sumOfPrimaryDiagonal=0, sumOfSecondaryDiagonal=0;

for (int row=0; row<theArray.length; row++){
    sumOfPrimaryDiagonal += theArray[row][row];
    sumOfSecondaryDiagonal += theArray[row][theArray.length-
   row-1];
}

System.out.println("sumOfPrimaryDiagonal:" +
   sumOfPrimaryDiagonal);
System.out.println("sumOfSecondaryDiagonal:" +
   sumOfSecondaryDiagonal);

int sum = sumOfPrimaryDiagonal + sumOfSecondaryDiagonal;

if (theArray.length%2 != 0)
    sum -= theArray[theArray.length/2][theArray.length/2];

System.out.println("total sum:" + sum);
```

Question 9: Write a program that compares two arrays of characters (not Strings!) of the same size and determines if they contain the same elements (but not necessarily in the same order).

For example, the following arrays contain the same elements:

'a'	'c'	'k'	'i'	'b'
'b'	'i'	'k'	'a'	'c'

so your program should display `true`.

One possible solution:

```
class CompareArrays {

  public static void main(String args[]) {

char[] array3 = {'a','c','k','i','b'};
char[] array2 = {'b','i','k','a','c'};
char[] array1 = {'b','i','a','a','c'};

// don't waste my time!
if (array1.length != array2.length) {
   System.out.print("false");
   System.exit(0);
}
boolean sameFrequency =  true;

for (int i=0; (i<array1.length && sameFrequency);  i++)
{
   int frequencyOfArray_i_InArray1 = 0;
   for (int j=0; j<array1.length; j++)
      if (array1[i] == array1[j])
         frequencyOfArray_i_InArray1++;

   int frequencyOfArray_i_InArray2 = 0;
   for (int j=0; j<array2.length; j++)
      if (array1[i] == array2[j])
         frequencyOfArray_i_InArray2++;

   if (frequencyOfArray_i_InArray2 != frequencyOfArray_i_InArray1)
      sameFrequency = false;
}

if (!sameFrequency)
   System.out.print("false");
else
   System.out.print("true");
   }
}
```

Question 10: A magic squares is an N-by-N matrix of the integers, such that all row, column, and diagonal sums are equal. For example,

```
4   9   2
3   5   7
8   1   6
```

is a magic square, because 4+9+2=15 3+5+7=15 8+1+6= 15 and 4+38=15 9+5+1=15 2+7+6=15 and 4+5+6=15 2+5+8 = 15.

Write a Java program to test if a matrix represents a magic square.

One possible solution:

```
       class testMagicSquare
       {
           public static void main (String[] arg)
           {
              int[][] theArray ={{4,9,2},
                                 {3,5,7},
                                 {8,1,6}};

               final int n = theArray.length;
               int previousSum = 0;

               int sumOfRow=0, sumOfColumn=0;
               int sumOfPrimaryDiagonal=0, sumOfSecondaryDiagonal=0;

               boolean allEqual = true;

               for (int row=0; (row<n && allEqual); row++){
                   sumOfRow=0;
                   sumOfColumn=0;
                   for (int col=0; col<n; col++){
                       sumOfRow += theArray[row][col];
                       sumOfColumn += theArray[col][row];
                   }

                   sumOfPrimaryDiagonal += theArray[row][row];
                   sumOfSecondaryDiagonal += theArray[row][n-row-1];

                   allEqual = (sumOfRow == sumOfColumn);

                    // it is not the first sum we calculate, so the
       current sum must also be
                  // equal to the previous sums
                   if (previousSum != 0)
                    allEqual = allEqual && (sumOfRow == previousSum);

                   previousSum = sumOfRow;
               }

               if (allEqual &&
                   previousSum == sumOfPrimaryDiagonal &&
                   previousSum == sumOfPrimaryDiagonal)
             System.out.println("Magic Square");
              else
             System.out.println("Not a magic Square");
           }
}
```

Question 11: One way to generate a magic square of size n, when n is odd is to assign the integers 1 to n^2 in ascending order, starting at the bottom, middle cell. Repeatedly assign the next integer to the cell adjacent diagonally to the right and down. If this cell has already been assigned another integer, instead use the cell adjacently above. If the new column is outside the square start back at the first column. If the new row is outside the square, start back at the beginning of the row.

Write a Java program to generate a magic square of a given odd size.

For example, if the user enters 3, you should generate:

```
4   9   2
3   5   7
8   1   6
```

if the user enters 5, you should generate:

```
11 18 25  2  9
10 12 19 21  3
 4  6 13 20 22
23  5  7 14 16
17 24  1  8 15
```

One possible solution:

```
import java.util.*;

public class GenerateMagicSquare {

    public static void main(String[] args) {
    System.out.print("Enter an odd integer: ");
        Scanner keyboard = new Scanner(System.in);
        int n = keyboard.nextInt();
        if (n % 2 == 0)
        {
          System.out.print("n must be odd");
          System.exit(0);
        }

        int[][] magic = new int[n][n];

        // set up first value at the bottom row in the middle
        int row = n-1;
        int col = n/2;
        magic[row][col] = 1;

        // set up all other integers from 2 to n*n
        for (int i = 2; i <= n*n; i++) {
          // try to go down one row (wrap around if needed)
          // and try to go right (wrap around if needed)
          // if cell has not been assigned yet
            if (magic[(row + 1) % n][(col + 1) % n] == 0) {
                row = (row + 1) % n;  // go down one row (wrap around if
needed)
                col = (col + 1) % n;  // go right (wrap around if needed)
            }
           // if cell has already been assigned, go up 1 row
            else {
                row = (row - 1 + n) % n;
                // don't change col
            }
            magic[row][col] = i;
        }

        // print results
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (magic[i][j] < 10)  System.out.print(" ");  // for
alignment
                if (magic[i][j] < 100) System.out.print(" ");  // for
alignment
                System.out.print(magic[i][j] + " ");
            }

            System.out.println();
        }
    }
}
```