Exam #4 Solutions

1 Short Answer (10 Points Each)

1. Write a declaration for a two dimensional array of doubles that has 14 rows and 17 columns. Then write a nested for loop that populates the array where each cell is the sum of the row index and the column index of the cell.

   Solution:

```
double[][] A = new double[14][17];

for (int i = 0; i < 14; i++)
    for (int j = 0; j < 17; j++)
        A[i][j] = i + j;
```

2. Write a segment of code that will find the largest entry in a two-dimensional array of doubles, `A`. Have the segment of code store the row and column positions of the largest entry. You do not know the size of the array before the block of code, nor do you know anything about the size of the entries in the array.

   Solution:

```
double max = A[0][0];
int maxrow = 0;
int maxcol = 0;
for (int i = 0; i < A.length; i++)
    for (int j = 0; j < A[0].length; j++)
        if (max < A[i][j]) {
            max = A[i][j];
            maxrow = i;
            maxcol = j;
        }
```

3. Write a segment of code that will shuffle a one-dimensional array of integers. The shuffle should be an interchange shuffle that interchanges two array values at random. The number of interchanges the block must do is five times the length of the array. You do not know the size of the array before this block of code is executed.

   Solution:

```
for (int i = 0; i < A.length * 5; i++) {
    int pos1 = (int) (Math.random() * A.length);
    int pos2 = (int) (Math.random() * A.length);
    int temp = A[pos1];
    A[pos1] = A[pos2];
    A[pos2] = temp;
}
```

4. Write a segment of code that will create an ArrayList that will store doubles and then populate the array with the numbers, 0, 2π, 4π, 6π, 8π, 10π, 12π, 14π, 16π, 18π, 20π, in that order. Finally print out the ArrayList in a single line.

   Solution:

```
ArrayList<Double> A = new ArrayList<Double>();
for (int i = 0; i <= 20; i += 2)
    A.add(i * Math.PI);

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

5. Write a segment of code that will add up all of the entries on the diagonal of a two-dimensional integer array, `A`. The diagonal of an array are the entries in the $(0,0)$, $(1,1)$, $(2,2)$, $\ldots$, (n,n) positions, where the (n,n) position is the largest possible position that is still in the array. So for a 4×9 array $n = 4$ and for a 5×3 array $n = 3$. You do not know the size of the array before this block of code is executed.

   Solution:

```
int n = A.length;
if (A[0].length < n)
    n = A[0].length;

int sum = 0;
for (int i = 0; i < n; i++)
    sum += A[i][i];
```

2 Program Traces (15 Points Each)

1. For each of the three runs, give the program output.

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class Exam04Trace1 {
    public static void alter(ArrayList<Integer> A)
         {
        int temp = A.get(0);
        A.set(0, A.get(A.size() - 1));
        A.set(A.size() - 1, temp);

        for (int i = 1; i < A.size() - 1; i += 2)
            A.set(i, i * A.get(i));

        System.out.println("One");
    }

    public static void alter(ArrayList<Integer> A,
         int n) {
        for (int i = 0; i < A.size() - 1; i++)
            A.set(i, n * A.get(A.size() - 1 - i));

        System.out.println("Two");
    }

    public static void alter(ArrayList<Integer> A,
         int n, int m) {
        if (n < 1 || n >= A.size())
            n = 1;

        if (m < 1 || m >= A.size())
            m = 1;

        int temp = A.get(n);
        A.set(n, A.get(m));
        A.set(m, temp);

        for (int i = 0; i < A.size(); i++)
            A.set(i, A.get((n*i + m) % A.size()));

        System.out.println("Three");
    }

    public static void main(String[] args) {
        ArrayList<Integer> A = new ArrayList<
             Integer>();

        Scanner keyboard = new Scanner(System.in);
        System.out.print("Input: ");
        int entries = keyboard.nextInt();

        for (int i = 0; i < entries; i++) {
            A.add(keyboard.nextInt());
        }
        System.out.println(A);

        if (A.get(1) > 10)
            alter(A);
        else if (A.get(1) > 5)
            alter(A, A.get(0));
        else
            alter(A, A.get(0), A.get(1));

        System.out.println(A);
        Collections.sort(A);
        System.out.println(A);
    }
}
```

Run #1 Solution

```
Input: 5 2 4 6 8 10 20 30
[2, 4, 6, 8, 10]
Three
[6, 4, 8, 6, 8]
[4, 6, 6, 8, 8]
```

Run #2 Solution

```
Input: 10 20 15 10 5 0 1 2 3 4 5
[20, 15, 10, 5, 0, 1, 2, 3, 4, 5]
One
[5, 15, 10, 15, 0, 5, 2, 21, 4, 20]
[0, 2, 4, 5, 5, 10, 15, 15, 20, 21]
```

Run #3 Solution

```
Input: 6 20 8 5 4 3 2
[20, 8, 5, 4, 3, 2]
Two
[40, 60, 80, 1600, 1200, 2]
[2, 40, 60, 80, 1200, 1600]
```

2. Given the Thing class and the main program below, write the output of the program on the next page. Make sure that you mark all blank spaces with ␣.

```java
public class Thing {
    private int num1;
    private double num2;
    private String str;

    public Thing(int i, double d, String s) {
        num1 = i;
        num2 = d;
        str = s;
    }

    public Thing(int i, double d) {
        num1 = i;
        num2 = d;
        str = "No String";
    }

    public Thing(int i) {
        num1 = i;
        num2 = 3.14159;
        str = "Approximation to Pi";
    }

    public Thing() {
        num1 = 5;
        num2 = 2.71828;
        str = "Approximation to Natural Base";
    }

    public int getInt() {
        return num1;
    }

    public double getDouble() {
        return num2;
    }

    public String getString() {
        return str;
    }

    public String toString() {
        String retstr = str + "\n";
        retstr += num1 + "\n";
        retstr += num2 + "\n";
        return retstr;
    }

    public void doSomething() {
        str = str.substring(0, str.length()/2);
    }

    public void doSomething(int i) {
        str = str.substring(i);
    }

    public void doSomething(int i, String s) {
        num1 = str.indexOf(s, i);
    }

    public void up() {
        num1++;
        num2 += num1;
    }

    public void down() {
        num1--;
        num2 -= num1;
    }

    public void three() {
        str += str + str + str;
        num1 = str.length();
        num2 = str.length() / 2;
    }
}
```

```java
public class Exam04Trace2 {

    public static void printdiv() {
        System.out.println("-----");
    }

    public static void main(String[] args) {
        Thing th1=new Thing(4,7.8,"Easy Trace");
        System.out.println(th1);

        Thing th2 = new Thing();
        System.out.println(th2);

        Thing th3 = new Thing(-7, 0.022);
        System.out.println(th3);

        Thing th4 = new Thing(2);
        System.out.println(th4);

        printdiv();

        th3.up();
        System.out.println(th3);

        th3.down();
        System.out.println(th3);

        printdiv();

        th2.doSomething(3, "to");
        System.out.println(th2);

        th2.doSomething();
        System.out.println(th2);

        printdiv();

        th1.doSomething(2, "T");
        th1.doSomething(th1.getInt());
        System.out.println(th1);

        th1.three();
        System.out.println(th1);

        printdiv();

        th4 = th2;
        System.out.println(th2);
        System.out.println(th4);

        th2.up();

        System.out.println(th2);
        System.out.println(th4);
    }
}
```

Program Output

```
Easy␣Trace
4
7.8

Approximation␣to␣Natural␣Base
5
2.71828

No␣String
-7
0.022

Approximation␣to␣Pi
2
3.14159

-----
No␣String
-6
-5.978

No␣String
-7
1.022

-----
Approximation␣to␣Natural␣Base
14
2.71828

Approximation␣
14
2.71828

-----
Trace
5
7.8

TraceTraceTraceTrace
20
10.0

-----
Approximation␣
14
2.71828

Approximation␣
14
2.71828

Approximation␣
15
17.71828

Approximation␣
15
17.71828
```

3 Coding (15 Points Each)

1. The main program and output from the program are below. Write the three methods, print, populate, and transpose for the program. The print method is to take in as a parameter only a single two dimensional array of integers, and print the contents of the array in columns that line up and use 6 spaces for each entry. The populate method is to take in as a parameter only a single two dimensional array of integers, and fill the array with random integers between 1 and 100. The transpose method is to take in as a parameter only a single two dimensional array of integers, and output another two dimensional array that transposes the array. The transpose of an array is where each row becomes a column and each column a row. That is, the first row of the original array is the first column of the transpose, and so on. With the transpose method, the original array is not to be altered.

```java
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("Rows: ");
    int rows = keyboard.nextInt();
    System.out.print("Columns: ");
    int cols = keyboard.nextInt();

    int[][] A = new int[rows][cols];
    populate(A);
    print(A);

    int[][] B = transpose(A);
    System.out.println();
    print(B);
}
```

Program Output

```
Rows: 3
Columns: 5
    79    36     7    94    52
    39    44    39    28    70
    93    33    74     9    62

    79    39    93
    36    44    33
     7    39    74
    94    28     9
    52    70    62
```

Solution:

```java
import java.util.Scanner;

public class Exam04Code1 {

    public static void print(int[][] A) {
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A[0].length; j++)
                System.out.printf("%6d", A[i][j]);
            System.out.println();
        }
    }

    public static void populate(int[][] A) {
        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                A[i][j] = (int) (Math.random() *
                    100 + 1);
    }

    public static int[][] transpose(int[][] A) {
        int[][] retarr = new int[A[0].length][A.
            length];

        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                retarr[j][i] = A[i][j];

        return retarr;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Rows: ");
        int rows = keyboard.nextInt();
        System.out.print("Columns: ");
        int cols = keyboard.nextInt();

        int[][] A = new int[rows][cols];
        populate(A);
        print(A);

        int[][] B = transpose(A);
        System.out.println();
        print(B);
    }
}
```

2. The main program and output from the program are below. Write the three methods, ColumnSums, and RowSums for the program. The print method is to take in as a parameter only a single one dimensional array of integers, and print the contents of the array on one line that uses 6 spaces for each entry. The ColumnSums method is to take in as a parameter only a single two dimensional array of integers, and return a one dimensional array that contains the column sums of the array. The RowSums method is to take in as a parameter only a single two dimensional array of integers, and return a one dimensional array that contains the row sums of the array. You do not need to write the populate method or the print method for the two dimensional arrays. None of the methods you create are to alter the contents of the input arrays.

```java
public static void main(String[] args) {
    Scanner keyboard = new Scanner(System.in);
    System.out.print("Rows: ");
    int rows = keyboard.nextInt();
    System.out.print("Columns: ");
    int cols = keyboard.nextInt();

    int[][] A = new int[rows][cols];
    populate(A);
    print(A);

    int[] colsums = ColumnSums(A);
    System.out.println();
    print(colsums);

    int[] rowsums = RowSums(A);
    System.out.println();
    print(rowsums);
}
```

Program Output

```
Rows: 5
Columns: 4
    77    84    95    41
    43    49    67    48
     8    52    70    90
    66    39    70    86
    48    11    97    94

   242   235   399   359

   297   207   220   261   250
```

Solution:

```java
import java.util.Scanner;

public class Exam04Code2 {

    public static void print(int[][] A) {
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < A[0].length; j++)
                System.out.printf("%6d", A[i][j]);
            System.out.println();
        }
    }

    public static void print(int[] A) {
        for (int i = 0; i < A.length; i++) {
            System.out.printf("%6d", A[i]);
        }
        System.out.println();
    }

    public static void populate(int[][] A) {
        for (int i = 0; i < A.length; i++)
            for (int j = 0; j < A[0].length; j++)
                A[i][j] = (int) (Math.random() *
                    100 + 1);
    }

    public static int[] ColumnSums(int[][] A) {
        int[] retarr = new int[A[0].length];

        for (int i = 0; i < A[0].length; i++){
            int sum = 0;
            for (int j = 0; j < A.length; j++)
                sum += A[j][i];

            retarr[i] = sum;
        }

        return retarr;
    }

    public static int[] RowSums(int[][] A) {
        int[] retarr = new int[A.length];

        for (int i = 0; i < A.length; i++){
            int sum = 0;
            for (int j = 0; j < A[0].length; j++)
                sum += A[i][j];

            retarr[i] = sum;
        }

        return retarr;
    }

    public static void main(String[] args) {
        Scanner keyboard = new Scanner(System.in);
        System.out.print("Rows: ");
        int rows = keyboard.nextInt();
        System.out.print("Columns: ");
        int cols = keyboard.nextInt();

        int[][] A = new int[rows][cols];
        populate(A);
        print(A);

        int[] colsums = ColumnSums(A);
        System.out.println();
        print(colsums);

        int[] rowsums = RowSums(A);
        System.out.println();
        print(rowsums);
    }
}
```