CSE 1001: Introduction to Computer Programming

Programming Assignment-VIII

(Multidimensional Arrays)

1. Write a java program to print *M*-by-*N* array in the tabular format.

2. Write a method that returns the sum of all the elements in a specified column in a matrix using the following header:

   ```
   public static double sumColumn(double[][] m, int columnIndex)
   ```

   Write a java program that reads a 3-by-4 matrix and displays the sum of each column. Here is a sample run:

   ```
   Enter a 3-by-4 matrix row by row:
   1.5 2 3 4
   5.5 6 7 8
   9.5 1 3 1
   Sum of the elements at column 0 is 16.5
   Sum of the elements at column 1 is 9.0
   Sum of the elements at column 2 is 13.0
   Sum of the elements at column 3 is 13.0
   ```

3. Write a method that sums all the numbers in the major diagonal in an $n * n$ matrix of double values using the following header:

   ```
   public static double sumMajorDiagonal(double[][] m)
   ```

   Write a java program that reads a 4-by-4 matrix and displays the sum of all its elements on the major diagonal. Here is a sample run:

   ```
   Enter a 4-by-4 matrix row by row:
   1 2 3 4.0
   5 6.5 7 8
   9 10 11 12
   13 14 15 16
   Sum of the elements in the major diagonal is 34.5
   ```

4. Write a java program to create a two-dimensional array b[][] that is a copy of an existing two-dimensional array a[][], under each of the following assumptions:

   ```
   a. a[][] is square
   b. a[][] is rectangular
   c. a[][] may be ragged
   ```

5. Suppose a teacher with *M* students and *N* Marks of each student is maintained in an (*M*+1)-by-(*N*+1) array, reserving the last column for each student's average mark and the last row for average test mark. Write a java program to compute the average mark for each student (average values of each row) and calculate the average test mark (average values of each column).

6. Write a method to add two matrices. The header of the method is as follows:

   ```
   public static double[][] addMatrix(double[][] a, double[][] b)
   ```

   In order to be added, the two matrices must have the same dimensions and the same or compatible types of elements. Let c be the resulting matrix. Each element c_{ij} is $a_{ij} + b_{ij}$. For example, for two 3 * 3 matrices a and b, c is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \end{pmatrix}$$

   Write a java program that prompts the user to enter two 3 * 3 matrices and displays their sum. Here is a sample run:

   ```
   Enter matrix1: 1 2 3 4 5 6 7 8 9
   Enter matrix2: 0 2 4 1 4.5 2.2 1.1 4.3 5.2
   The matrices are added as follows
   1.0 2.0 3.0   0.0 2.0 4.0   1.0 4.0 7.0
   4.0 5.0 6.0 + 1.0 4.5 2.2 = 5.0 9.5 8.2
   7.0 8.0 9.0   1.1 4.3 5.2   8.1 12.3 14.2
   ```

7. Write a java program to transpose a square two-dimensional array in place without creating a second array.

8. Write a java program that takes an integer N from the command line and creates an N-by-N boolean array a[][] such that a[i][j] is true if i and j are relatively prime (have no common factors), and false otherwise.

9. Write a method to multiply two matrices. The header of the method is:

   ```
   public static double[][] multiplyMatrix(double[][] a, double[][] b)
   ```

   To multiply matrix **a** by matrix **b**, the number of columns in **a** must be the same as the number of rows in **b**, and the two matrices must have elements of the same or compatible types. Let **c** be the result of the multiplication. Assume the column size of matrix **a** is **n**. Each element c_{ij} is $a_{i1} * b_{1j} + a_{i2} * b_{2j} + \ldots. + a_{in} * b_{nj}$

   For example, for two 3 * 3 matrices **a** and **b, c** is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

   where $c_{ij} = a_{i1} * b_{1j} + a_{i2} * b_{2j} + a_{i3} * b_{3j}$.

Write a java program that prompts the user to enter two 3 * 3 matrices and displays their product. Here is a sample run:

```
Enter matrix1: 1 2 3 4 5 6 7 8 9
Enter matrix2: 0 2 4 1 4.5 2.2 1.1 4.3 5.2
The multiplication of the matrices is
1 2 3   0 2.0 4.0    5.3  23.9 24
4 5 6 * 1 4.5 2.2  = 11.6 56.3 58.2
7 8 9   1.1 4.3 5.2  17.9 88.7 92.4
```

10. Write a java program that randomly fills in 0s and 1s into a 4-by-4 matrix, prints the matrix, and finds the first row and column with the most 1s. Here is a sample run of the program:

```
0011
0011
1101
1010

The largest row index: 2
The largest column index: 2
```

****

Homework (Miscellaneous Questions on Arrays)

1. Write a java program to find the maximum and minimum and how many times they both occur in an array of *n* elements. Find out the positions where the maximum first occurs and the minimum last occurs.

2. Write a java program that takes two command-line arguments *M* and *N* and produces a sample of *M* of the integers from 0 to *N-1*.

3. Write a java program that simulates coupon collection by taking a command-line argument *N* and generating random numbers between 0 to *N*-1 until getting every possible value.

4. Write a java program that takes a command line argument *N* and computes the number of primes less than or equal to *N* using the algorithm of "*Sieve of Eratosthenes*".

5. Given an array of *N* elements with each element between 1 and *N*, write a java program to determine whether there are any duplicates. You do not need to preserve the contents of the given array, but do not use an extra array.

6. Write two overloaded methods that return the average of an array with the following headers:

   ```
   public static int average(int[] array)
   public static double average(double[] array)
   ```

   Write a java program that prompts the user to enter ten double values, invokes this method, and displays the average value.

7. Write a method that returns a sorted string using the following header:

   ```
   public static String sort(String s)
   ```

   For example, sort("acb") returns abc. Write a java program that prompts the user to enter a string and displays the sorted string.

8. Given an array of integers. Write a java program to find the length and location of the longest contiguous sequence of equal values where the values of the elements just before and just after this sequence are smaller.

9. You set your music player to shuffle mode. It plays each of the *N* songs before repeating any. Write a java program to estimate the likelihood that you will not hear any sequential pair of songs (that is, song 3 does not follow song 2, song 10 does not follow song 9, and so on).

10. Write a java program that reads in a permutation of the integers 0 to *N*-1 from *N* command-line arguments and prints the inverse permutation. (If the permutation is in an array a[], its inverse is the array b[] such that a[b[i]] = b[a[i]] = i.) Be sure to check that the input is a valid permutation.

11. There are 500 light bulbs (numbered 1 to 500) arranged in a row. Initially they are all OFF. Starting with bulb 2, all even numbered bulbs are turned ON. Next, starting with bulb 3, and visiting every third bulb, it is turned ON if it is OFF, and it is turned OFF if it is ON. This procedure is repeated for every fourth bulb, then every fifth bulb, and so on up to the 500th bulb. Write a java program to determine which bulbs are OFF at the end of above exercise.

12. Write a java program that computes the product of two square matrices of boolean values, using the *or* operation instead of + and the *and* operation instead of *.

    .

13. Write the following method that returns the location of the largest element in a two-dimensional array.

    ```
    public static int[] locateLargest(double[][] a)
    ```

    The return value is a one-dimensional array that contains two elements. These two elements indicate the row and column indices of the largest element in the two-dimensional array. Write a java program that prompts the user to enter a two dimensional array and displays the location of the largest element in the array. Here is a sample run:

    ```
    Enter the number of rows and columns of the array: 3 4
    Enter the array:
    23.5 35 2 10
    4.5 3 45 3.5
    35 44 5.5 9.6
    The location of the largest element is at (1, 2)
    ```

14. Write a method to sort a two-dimensional array using the following header:

    ```
    public static void sort(int m[][])
    ```

    The method performs a primary sort on rows and a secondary sort on columns. For example, the following array

    ```
    {{4, 2},{1, 7},{4, 5},{1, 2},{1, 1},{4, 1}}
    ```

    will be sorted to

    ```
    {{1, 1},{1, 2},{1, 7},{4, 1},{4, 2},{4, 5}}.
    ```

15. An *n* * *n* matrix is called a *positive Markov matrix* if each element is positive and the sum of the elements in each column is 1. Write the following method to check whether a matrix is a Markov matrix.

    ```
    public static boolean isMarkovMatrix(double[][] m)
    ```

    Write a java program that prompts the user to enter a 3 * 3 matrix of double values and tests whether it is a Markov matrix. Here are sample runs:

    ```
    Enter a 3-by-3 matrix row by row:
    0.15 0.875 0.375
    0.55 0.005 0.225
    0.30 0.12  0.4
    It is a Markov matrix

    Enter a 3-by-3 matrix row by row:
    0.95  -0.875  0.375
    0.65  0.005   0.225
    0.30  0.22    -0.4
    It is not a Markov matrix
    ```

16. Implement the following method to sort the rows in a two dimensional array. A new array is returned and the original array is intact.

```
public static double[][] sortRows(double[][] m)
```

Write a java program that prompts the user to enter a 3 * 3 matrix of double values and displays a new row-sorted matrix. Here is a sample run:

```
Enter a 3-by-3 matrix row by row:
0.15 0.875 0.375
0.55 0.005 0.225
0.30 0.12  0.4

The row-sorted array is
0.15  0.375 0.875
0.005 0.225 0.55
0.12  0.30  0.4
```

17. Implement the following method to sort the columns in a two-dimensional array. A new array is returned and the original array is intact.

```
public static double[][] sortColumns(double[][] m)
```

Write a java program that prompts the user to enter a 3 * 3 matrix of double values and displays a new column-sorted matrix. Here is a sample run:

```
Enter a 3-by-3 matrix row by row:
0.15 0.875 0.375
0.55 0.005 0.225
0.30 0.12  0.4

The column-sorted array is
0.15 0.0050 0.225
0.3  0.12   0.375
0.55 0.875  0.4
```

****