Assignment . **3**

Muhammad **Mujtaba** SP22-BSE-036

Sir **Rizwan** Rashid

Date: Dec 17, 2022

CODE

**Questions** highlighted with yellow, **methods** with green. Output at the end.  
  
*// - START  
  
// Mujtaba SP22-BSE-036  
// Sir Rizwan Rashid  
// Assignment . 3  
// Programming Fundamentals  
  
// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >*import java.io.\*; import java.util.\*;  
public class Assignment3 {  
 *// QUESTION . 1* void static matXmat() {  
 int a[][] = {{14, 41, 17}, {25, 2, 32}, {563, 73, 23}};  
 int b[][] = {{1043, 1111, 321}, {224, 522, 263}, {53, 32, 63}};  
 int c[][] = new int[3][3];  
 *//multiplying and printing multiplication of 2 matrices* for (int i = 0; i < 3; i++) {  
 for (int j = 0; j < 3; j++) {  
 c[i][j] = 0;  
 for (int k = 0; k < 3; k++){ c[i][j] += a[i][k] \* b[k][j]; }  
 System.out.print(c[i][j] + " "); *//printing matrix element* }  
 System.out.println();*//new line* }  
 }  
 *// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // QUESTION . 2* void static matrixFillFind(){  
 Scanner input = new Scanner(System.in);  
 System.out.print("Enter the size for the matrix: ");  
 int intSize = input.nextInt();  
  
 int[][] intMetrix = new int[intSize][intSize];  
 fillMetrix(intMetrix);  
 displayMetrix(intMetrix);  
 checkMetrix(intMetrix);  
 }  
 public static int intRandom(int lowerBound, int upperBound) {  
 return (int) (lowerBound + Math.random()  
 \* (upperBound - lowerBound + 1));  
 }  
 public static void fillMetrix(int metrix[][]) {  
 for (int i = 0; i < metrix.length; i++) {  
 for (int j = 0; j < metrix[i].length; j++) {  
 metrix[i][j] = intRandom(0, 1);  
 }  
 }  
 }  
 public static void checkMetrix(int metrix[][]) {  
 boolean blnFound = false;  
 *// Checking row* for (int i = 0; i < metrix.length; ++i) {  
 int intResult = checkRow(i, metrix);  
 if (intResult != 2) {  
 blnFound = true;  
 System.out.println("All " + intResult + "s on row " + i);  
 }  
 }  
 if (blnFound == false)System.out.println("No same numbers on a row");  
 *// Checking column* blnFound = false;  
 for (int i = 0; i < metrix.length; ++i) {  
 int intResult = checkCol(i, metrix);  
 if (intResult != 2) {  
 blnFound = true;  
 System.out.println("All " + intResult + "s on column " + i);  
 }  
 }  
  
 if (blnFound == false) System.out.println("No same numbers on a column");  
  
 *// Checking major diagonals* int intMajor = checkMajorDia(metrix);  
 int intMinor = checkMinorDia(metrix);  
  
 if (intMajor != 2) {  
 System.out.println("All " + intMajor + "s on major diagonal");  
 } else {  
 System.out.println("No same numbers on the major diagonal");  
 }  
  
 if (intMinor != 2) {  
 System.out.println("All " + intMinor + "s on sub-diagonal");  
 } else {  
 System.out.println("No same numbers on the sub-diagonal");  
 }  
  
 }  
 public static void displayMetrix(int metrix[][]) {  
 for (int i = 0; i < metrix.length; i++) {  
 for (int j = 0; j < metrix[i].length; j++) {  
 System.out.print(metrix[i][j]);  
 } System.out.println("");  
 }  
 }  
  
 *// Return 2 if the row is not repeating. Return number 0 or one if it is* public static int checkRow(int intRow, int intMetrix[][]) {  
 for (int i = 1; i < intMetrix[intRow].length; i++) {  
 if (intMetrix[intRow][i] != intMetrix[intRow][0])  
 return 2;  
 }  
 return intMetrix[intRow][0];  
 }  
 public static int checkCol(int intCol, int intMetrix[][]) {  
 for (int i = 1; i < intMetrix.length; i++) {  
 if (intMetrix[i][intCol] != intMetrix[0][intCol])  
 return 2;  
 }  
 return intMetrix[0][intCol];  
 }  
 *// Check major diagonal where i = j or from top left to lower right* public static int checkMajorDia(int intMetrix[][]) {  
 for (int i = 1; i < intMetrix.length; i++) {  
 if (intMetrix[i][i] != intMetrix[0][0])  
 return 2;  
 }  
 return intMetrix[0][0];  
 }  
 *// Check major diagonal where i = metrix size - i or from lower left to  
 // upper right* public static int checkMinorDia(int intMetrix[][]) {  
 int intSize = intMetrix.length - 1;  
 for (int i = 1; i < intMetrix.length; i++) {  
 if (intMetrix[i][intSize - i] != intMetrix[0][intSize])  
 return 2;  
 }  
 return intMetrix[0][0];  
 }  
 *// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // QUESTION . 3* public static void sortByOrder() {  
 char[] arr = {'c', 'a', 'b', 'a', 'a', 'b', 'a', 'a', 'c', 'a'};  
 int aC = 0;  
 int bC = 0, int cC = 0;  
 for (int i = 0; i < arr.length(); i++) {  
 if (arr[i] == 'a') aC++;  
 if (arr[i] == 'b') bC++;  
 if (arr[i] == 'c') cC++;  
 }  
 for (int i = 0; i < aC + 1; i++) arr[i] = 'a';  
 for (int i = 0; i < bC + 1; i++) arr[i] = 'b';  
 for (int i = 0; i < cC + 1; i++) arr[i] = 'c';  
  
 }  
 *// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // QUESTION . 4* public static void question4() {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Enter the size of the array you want to create: ");  
 int size = input.nextInt();  
 int arr[] = new int[size];  
 int x = 0;  
 for (int element : arr) {  
 System.out.print("Enter the digit you want to enter in the array: ");  
 int num = input.nextInt();  
 arr[x] = num;  
 x++;  
 }  
 System.out.print("Choose any of the following options:\n1) count\n2) Partition\n3) Duplicate\n4) Circular\n5)Shift Circular");  
 int c = input.nextInt();  
 if (c == 1) count(arr);  
 if (c == 2) partition(arr, size);  
 if (c == 3) duplicate(arr, size);  
 if (c == 4) circular(arr, size);  
 if (c == 5) shiftCircular(arr, size);  
 }  
  
 public static void count(int arr[]) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Enter the number whom occurence you want to find in array: ");  
 int num = input.nextInt();  
 int count = 0;  
 for (int element : arr) { if (element == num) count++; }  
 System.out.println("The number occurred " + count + " times.");  
 }  
  
 public static void partition(int arr[], int size) {  
 int num = arr[0];  
 int count\_greater = 0;  
 int count\_smaller = 0;  
  
 for (int element : arr) {  
 if (element > num) count\_greater++;  
 if (element < num) count\_smaller++;  
 }  
  
 int greater[] = new int[count\_greater];  
 int smaller[] = new int[count\_smaller];  
  
 int x = 0;  
 for (int a : arr) { if (a > num) { greater[x] = a; x++; } }  
 x = 0;  
 for (int a : arr){ if (a < num){ smaller[x] = a; x++;} }  
 x = 0;  
 if (size >= x) {  
 for (int a : smaller) { arr[x] = a; x++;}  
 if (num == num){ arr[x] = num;x++;}  
 for (int b : greater){ arr[x] = b; x++;}  
 }  
 for (int z : arr) System.out.println(z);  
 }  
 public static void circular(int arr[], int size) {  
 int x = 0;  
 int y = 0;  
 int newArr[] = new int[size];  
 int temp1 = arr[0];  
 int temp2 = arr[1];  
 int temp3 = arr[size - 1];  
 try {  
 for (int element : arr) {  
 if ((x + 1) != (size - 1) & x != (size - 1)) {  
 newArr[x] = arr[y + 1] + arr[y + 2];  
 y++; x++;  
 }  
 if (x + 1 == size - 1){ newArr[x] = temp3 + temp1; x++; }  
 if (x == size - 1){ newArr[x] = temp1 + temp2; }  
 }  
 } catch (Exception e) { System.out.println("A2"); }  
 for (int element : newArr) { System.out.println(element); }  
 }  
  
 public static void shiftCircular(int arr[], int size) {  
 int temp1 = arr[0]; int temp2 = arr[1]; int x = 0;  
 for (int element : arr) {  
 if (x != size - 1 & x != size - 2) {arr[x] = arr[x + 2];x++;}  
 if (x == size - 2) {arr[x] = temp1;x++;}  
 if (x == size - 1) arr[x] = temp2;  
 }  
 for (int element : arr) {System.out.println(element);}  
 }  
  
 public static void duplicate(int arr[], int size) {  
 int[] frequencies = new int[size];  
 int visited = -1;  
 for (int i = 0; i < size; i++){  
 int count = 1;  
 for (int j = i + 1; j < size; j++) {  
 if (arr[i] == arr[j]){ count++;frequencies[j] = visited; }  
 }  
 if (frequencies[i] != visited) frequencies[i] = count;  
 }  
 for (int i = 0; i < frequencies.length; i++){  
 if (i == 0) System.out.println("Element : Frequency");  
 if (frequencies[i] != visited) System.out.println(arr[i] + " : " + frequencies[i]);  
 }  
 }  
 *// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // QUESTION . 5* public static void shift(int n) {  
 int[] datas = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};  
 int[] temp = new int[datas.length];  
 for (int i = 0; i < temp.length; i++){  
 temp[i] = datas[(i + n) % temp.length];  
 }  
 datas = temp;  
 for(int i = 0; i < datas.length(); i++) System.out.println(datas[i]);  
 }  
 *// > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // > > > > > > > > > > > > > > > > > > > > > > > > > > > > > >  
 // MAIN (to test all questions)* public static void main(String[] args){  
 matXmat();  
 matrixFillFind();  
 sortByOrder();  
 question4();  
 shift(2);  
 }  
}  
 *// - END*

OUTPUT

