**ICS 102 Problem Set: 2D-Arrays**

**// Finds sum of even values in a 2D-array**

**public class SumEvenValues2DArray {**

**public static void main(String[] args) {**

**int[][] x = {{3, 2, 1}, {4, 6, 7}, {1, 1, 2}};**

**int sum = sumEven(x);**

**System.out.printf("Sum of even values = %d ", sum);**

**}**

**private static int sumEven(int[][] y){**

**int sum = 0;**

**for(int r = 0; r < y.length; r++){**

**for(int c = 0; c < y[r].length; c++){**

**if(y[r][c] % 2 == 0){**

**sum += y[r][c];**

**}**

**}**

**}**

**return sum;**

**}**

**}**

**// Finds the sum of even values in each row of a 2D-array**

**public class SumEvenValues2DArray2 {**

**public static void main(String[] args) {**

**int[][] x = {{3, 2, 1}, {4, 6, 7}, {1, 1, 2}};**

**int[] sums = sumEven(x);**

**System.out.printf("%9s%9s%n", "Row Index", "Sum");**

**for(int k = 0; k < sums.length; k++){**

**System.out.printf("%9d%9d%n", k, sums[k]);**

**}**

**}**

**private static int[] sumEven(int[][] y){**

**int rowSum = 0;**

**int[] sums = new int[y.length];**

**for(int r = 0; r < y.length; r++){**

**rowSum = 0;**

**for(int c = 0; c < y[r].length; c++){**

**if(y[r][c] % 2 == 0){**

**rowSum += y[r][c];**

**}**

**}**

**sums[r] = rowSum;**

**}**

**return sums;**

**}**

**}**

**// Finds the locations of even values in a 2D-array**

**public class LocationsEvenValues2DArray {**

**public static void main(String[] args) {**

**int[][] x = {{3, 2, 1}, {4, 6, 7}, {1, 1, 2}};**

**int[][] locations = getLocationEvens(x);**

**if(locations[0].length == 0)**

**System.out.printf("There are no even values in the array");**

**else if(locations[0].length == 1){**

**System.out.printf("The location of even value is: ");**

**System.out.printf("(%d, %d) ", locations[0][0], locations[1][0]);**

**}else{**

**System.out.printf("The locations of even values are: ");**

**for(int k = 0; k < locations[0].length; k++)**

**System.out.printf("(%d, %d) ", locations[0][k], locations[1][k]);**

**}**

**}**

**private static int[][] getLocationEvens(int[][] y){**

**int count = 0;**

**int[][] locations = new int[2][y.length \* y[0].length];**

**for(int r = 0; r < y.length; r++){**

**for(int c = 0; c < y[r].length; c++){**

**if(y[r][c] % 2 == 0){**

**locations[0][count] = r;**

**locations[1][count] = c;**

**count++;**

**}**

**}**

**}**

**if(count == y.length \* y[0].length)**

**return locations;**

**int[][] locations2 = new int[2][count];**

**for(int c = 0; c < count; c++){**

**locations2[0][c] = locations[0][c];**

**locations2[1][c] = locations[1][c];**

**}**

**return locations2;**

**}**

**}**

1. What is the output of the following program fragment?

int j, k ;

int[][] b = {{57,38,39},{40,41,42},{33,54,45}};

for (k = 0;k < 3;k += 2) {

for (j = 2;j >= 0;j--)

System.out.printf("%d\t",b[j][k]);

}

System.out.printf("\n");

for(k = 4;k < 7; k++)

System.out.printf("%d\t",b[k%2][k%3]);

1. What is the output of the program fragment below if the input is: **1 2 3 4 5 6 7 8 9** ?

int[][] a = new int[3][3];

System.out.printf("Enter 9 values: ");

for(int i=2;i>=0;i--) {

for (int j=i;j>=0;j--) {

if(i==j)

a[i][j] = scanner.nextInt();

else{

a[i][j] = scanner.nextInt();

a[j][i] = scanner.nextInt();

}

}

}

for(int i=0;i<3;i++){

for(int j=0;j<3;j++)

System.out.printf("%d\t",a[i][j]);

System.out.println();

}

1. What is the output of the following program fragment?

final int ROWS = 3, COLUMNS = 4;

int[][] z = {{5,10,9,11},{2,6,3,8},{1,7,4,12}};

int i,j,m;

for (j=0; j<COLUMNS;j++) {

m = 999;

for (i=0;i<ROWS;i++) {

if (z[i][j] < m)

m = z[i][j];

}

System.out.printf ("%d\t", m);

}

1. What is the output of the program fragment below if the input is: **5 3 1 7 8 6 15 11 2 -1 0 9** ?

int[][] a = new int[3][3];

int i, j;

for(j = 1; j <= 5; j = j + 2) {

for(i =j; i != j+3; i++) {

a[i%3][j%3] = scanner.nextInt();

}

}

for(i = 0; i < 3; i++) {

for(j = 0; j < 3; j++)

System.out.printf("%4d", a[i][j]);

System.out.printf("\n");

}

1. What will be the output of the following code fragment:

int[][] x ={{21,12,43},{24,55,76},{64,90,45}};

int i, j;

for(i = 2; i >= 0; i--)

for(j = 0; j < i; j++)

System.out.printf("%d ",x[i][j]);

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | A. | 12 55 90 |
|  | B. | 64 90 24 |
|  | C. | 21 12 24 |
|  | D. | 43 12 64 |

|  |  |  |
| --- | --- | --- |
| The next two questions (6 and 7) are based on the following code:  int[][] x ={{2,6,-7,1},{7,-4,3,5},{6,8,-9,5},{3,6,9,-1}};  int max,i;  **\_\_\_\_\_Statement 1\_\_\_\_\_**  for(i=1; i<=3; i++)  **\_\_\_\_\_Statement 2\_\_\_\_\_**  max = x[i][i];  The objective of the above code is to find the maximum value of the diagonal of array x and save it in variable max. What should be the missing 2 statements? | | |
| 6. | Statement 1 should be: | |
|  | A. | max = x[0][0]; |
|  | B. | max = x[0][4]; |
|  | C. | max = x[2][0]; |
|  | D. | max = x[0][3]; |
| 7. | Statement 2 should be: | |
|  | A. | if(x[i-1][i-1] < max) |
|  | B. | if(max > x[i+1][i+1]) |
|  | C. | if(x[i][i] > max) |
|  | D. | if(x[4][4] < max) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 8. | Which code fragment will assign the following values to array **x**? | | | |  |  |  |  | | --- | --- | --- | --- | | 0 | 9 | 9 | 9 | | 7 | 0 | 9 | 9 | | 7 | 7 | 0 | 9 | |
|  |  | | |
| A. | int i,j;  int x[3][4]={0};  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else  x[i][j]=9;  }} | B. | int i,j;  int[][] x= new int[3][4];  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else if (i< j)  x[i][j]=9;  }} | |
| C. | int i,j, x[3][4];  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i> j)  x[i][j]=7;  else if (i< j)  x[i][j]=9;  }} | D. | int i,j;  int[][] x= new int[3][4];  for (i=0; i<3; i++) {  for (j=0; j<4; j++) {  if(i>= j)  x[i][j]=7;  else  x[i][j]=9;  }} | |

9. Which code fragment will initialize array x as shown?

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |

|  |  |
| --- | --- |
| 1. int[][] x = new int[4][4];   for(int m = 0; m <= 3; m++)  for(int k = 3; k >= 0; k--)  x[m][k] = 6 - (k + m); | 1. int[][] x = new int[4][4];   for(int k = 0; k <= 3; k++)  for(int m = 0; m <= 3; m++)  x[k][m] = k + m; |
| 1. int[][] x = new int[4][4];   for(int k = 3; k >= 0; k--)  for(int m = 3; m >= 0; m--)  x[k][3- m] = k + m; | 1. int j = 0;   int[][] x = new int[4][4];  for(int k = 0; k <= 3; k++)  for(int m = 0; m <= 3; m++)  x[k][m] = j++; |

1. 1
2. 2
3. 3
4. 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 10. Show the values of matrix b resulting from executing the program  fragment shown below:  int b = new int[3][4];  for(int i = 0; i < 3; i++)  for(int j = 0; j < 4; j++)  if(i < 1)  b[i][j] = j+2;  else  b[i][j] = b[i-1][j]\*j; | |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |  |  |  |  | |  |  |  |  | |

11. Show the contents of array x after executing the following code:

int[][] x ={{5,6,8}, {2, 4,3},{7, 1, 9}};

int k, m, temp;

for(k = 1; k < 3; k++){

temp = x[3 - k][0];

x[3 - k][0] = x[0][k];

x[0][k] = temp;

}

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

12. Write a Java program that creates and initializes the two dimensional array as shown below using **nested**

**loops**.

|  |  |  |  |
| --- | --- | --- | --- |
| 0 | 1 | 1 | 1 |
| 2 | 0 | 1 | 1 |
| 2 | 2 | 0 | 1 |
| 2 | 2 | 2 | 0 |
| 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 |

13.Write a program that reads the values of a 3 \* 4 matrix row-wise. It then finds, column-wise, the location

of the maximum value i.e. its row index and its column index.

14. Write a Java program that declares a 2-D array of integer values with 5 rows and 4 columns and a 1-D array

sumcols to store the sum of each column of the 2-D array. Your program will do the following:

* Reads the values from a file “data.txt” row-wise (the file contains exactly 20 values) and store them in

the 2-D array.

* Computes the sum of each column of the 2-D array and store the results in the 1-d array sumcols in the

corresponding locations i.e. first element of sumcols contains sum of first column of 2-D array and so on.

* Prints the sum of each column.

1. Write a Java program that sums all rows and all columns of an integer matrix. The matrix must have one additional column to sum all rows and one additional row to sum all columns.

For example:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 8 | -2 | 5 |  |  | 4 | 8 | -2 | 5 | **15** |
| 1 | 3 | 0 | 6 |  | 1 | 3 | 0 | 6 | **10** |
| 7 | -1 | 9 | 2 |  | 7 | -1 | 9 | 2 | **17** |
|  |  |  |  |  | **12** | **10** | **7** | **13** | **42** |

1. Write a Java program that prompts for and reads an integer number *k*. It then prompts for and reads a 3 \* 4integer matrix *A* row-wise. The program finally tests whether the matrix contains a row whose sum of elements is equal to *k*.
2. Given a square matrix *x* of size ***n***, write a Java program to swap the ***first*** and ***last*** rows. DO NO use any additional array. The example below is for a matrix *x* of size *n*=3.

|  |  |
| --- | --- |
| ***x* before:**  11 12 13  24 25 26  37 38 39 | ***x* after:**  37 38 39  24 25 26  11 12 13 |

1. Given a square matrix *x* of size *n*, write a Java program to copy all the elements of *x* to 1-D array *y* of size *n\*n*. The copying is to be done row-wise. For example, if we use the matrix *x* given above (before swapping), then *y* = {11, 12, 13, 24, 25, 26, 37, 38, 39}.
2. Write a Java program that computes the sum of the elements in Column x of a 2D array (careful with rows of different lengths!)