Week 4

Task 3

```
import java.util.*;


public class Solution {


  static int[] compareTriplets(int[] a, int[] b) {
    int aliceScore = 0;
    int bobScore = 0;


    for (int i = 0; i < 3; i++) {
      if (a[i] > b[i]) {
        aliceScore++;
      } else if (a[i] < b[i]) {
        bobScore++;
      }
    }
    return new int[]{aliceScore, bobScore};
  }
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[] a = new int[3];
    int[] b = new int[3];
    for (int i = 0; i < 3; i++) {
      a[i] = sc.nextInt();
    }
    for (int i = 0; i < 3; i++) {
      b[i] = sc.nextInt();
```

```
        }

        int[] result = compareTriplets(a, b);
        System.out.println(result[0] + " " + result[1]);
        sc.close();
    }
}
```

Task 4:

```
class Solution {
    public boolean containsDuplicate(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            for (int j = i + 1; j < nums.length; j++) {
                if (nums[i] == nums[j]) {
                    return true;
                }
            }
        }
        return false;
    }
}
```

Task 5:

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
```

```
import java.util.regex.*;

class Result {

  /*
   * Complete the 'timeConversion' function below.
   *
   * The function is expected to return a STRING.
   * The function accepts STRING s as parameter.
   */

  public static String timeConversion(String s) {
    String period = s.substring(s.length() - 2);
    String time = s.substring(0, 8);
    String[] parts = time.split(":");
    int hour = Integer.parseInt(parts[0]);

    if (period.equals("AM")) {
      if (hour == 12) {
        hour = 0;
      }
    } else {
      if (hour != 12) {
        hour += 12;
      }
    }

    return String.format("%02d:%s:%s", hour, parts[1], parts[2]);
```

```
    }
}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        String s = bufferedReader.readLine();

        String result = Result.timeConversion(s);

        bufferedWriter.write(result);
        bufferedWriter.newLine();

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Task 6:

```
class Solution {
    public void moveZeroes(int[] nums) {
        int index = 0;

        for (int i = 0; i < nums.length; i++) {
            if (nums[i] != 0) {
```

```
            nums[index] = nums[i];
            index++;
        }
      }
      while (index < nums.length) {
        nums[index] = 0;
        index++;
      }
    }
}
```

Task 7:

```
import java.util.Scanner;

public class Solution {

    public static int diagonalDifference(int[][] arr) {
      int n = arr.length;
      int primarySum = 0;
      int secondarySum = 0;

      for (int i = 0; i < n; i++) {
        primarySum += arr[i][i];
        secondarySum += arr[i][n - 1 - i];
      }

      return Math.abs(primarySum - secondarySum);
    }
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[][] arr = new int[n][n];

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                arr[i][j] = sc.nextInt();
            }
        }

        System.out.println(diagonalDifference(arr));
        sc.close();
    }
}
```

Task 8:

```
class Solution {
    public int[][] transpose(int[][] matrix) {
        int i;
        int j;
        int m=matrix.length;
        int n=matrix[0].length;
        int result[][]=new int[n][m];
        for(i=0;i<m;i++){
            for(j=0;j<n;j++){
                result[j][i]=matrix[i][j];
```

```
        }
      }
      return result;

  }
}
```

Task 9:

```
class Solution {
   public int[][] matrixBlockSum(int[][] mat, int k) {
      int m = mat.length;
      int n = mat[0].length;
      int[][] pre = new int[m + 1][n + 1];
      for (int i = 1; i <= m; i++) {
         for (int j = 1; j <= n; j++) {
            pre[i][j] = mat[i - 1][j - 1]
                  + pre[i - 1][j]
                  + pre[i][j - 1]
                  - pre[i - 1][j - 1];
         }
      }

      int[][] answer = new int[m][n];
      for (int i = 0; i < m; i++) {
         for (int j = 0; j < n; j++) {

            int r1 = Math.max(0, i - k);
```

```
            int c1 = Math.max(0, j - k);
            int r2 = Math.min(m - 1, i + k);
            int c2 = Math.min(n - 1, j + k);

            answer[i][j] = pre[r2 + 1][c2 + 1]
                    - pre[r1][c2 + 1]
                    - pre[r2 + 1][c1]
                    + pre[r1][c1];
        }
      }

      return answer;
   }
}
```

Task 10:

```
import java.util.*;

public class Solution {

   public static void matrixRotation(int[][] matrix, int r) {
      int m = matrix.length;
      int n = matrix[0].length;

      int layers = Math.min(m, n) / 2;

      for (int layer = 0; layer < layers; layer++) {

         List<Integer> elements = new ArrayList<>();
```

```
int top = layer;
int left = layer;
int bottom = m - layer - 1;
int right = n - layer - 1;

for (int j = left; j <= right; j++)
   elements.add(matrix[top][j]);

for (int i = top + 1; i <= bottom - 1; i++)
   elements.add(matrix[i][right]);

for (int j = right; j >= left; j--)
   elements.add(matrix[bottom][j]);

for (int i = bottom - 1; i >= top + 1; i--)
   elements.add(matrix[i][left]);

int size = elements.size();
int rotations = r % size;

Collections.rotate(elements, -rotations);

int index = 0;
for (int j = left; j <= right; j++)
   matrix[top][j] = elements.get(index++);
for (int i = top + 1; i <= bottom - 1; i++)
   matrix[i][right] = elements.get(index++);
```

```
        for (int j = right; j >= left; j--)
            matrix[bottom][j] = elements.get(index++);
        for (int i = bottom - 1; i >= top + 1; i--)
            matrix[i][left] = elements.get(index++);
    }
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(matrix[i][j] + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int m = sc.nextInt();
    int n = sc.nextInt();
    int r = sc.nextInt();
    int[][] matrix = new int[m][n];

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            matrix[i][j] = sc.nextInt();
        }
    }
    matrixRotation(matrix, r);
    sc.close();
}
```

```
}
```