```java
public class AdvSubArray {
    public static int maxScore(int[] nums, int k) {
        int n = nums.length;
        int maxScore = Integer.MIN_VALUE;

        for (int i = 0; i <= n - k; i++) {
            int currentScore = 0;
            for (int j = 0; j < k; j++) {
                currentScore += (j + 1) * nums[i + j];
            }
            maxScore = Math.max(maxScore, currentScore);
        }

        return maxScore;
    }
    public static void main(String[] args) {
        int[] distances = {2, 3, 1, 5, 4};
        int k = 3;
        int result = maxScore(distances, k);
        System.out.println("Maximum possible score: " + result);

    }
}

//Maximum possible score: 23
```

```java
public class AntOnRail {
    public static void main(String[] args) {
        int[] moves = {1, -1, 1, -1, 1};
        int numberOfMoves = moves.length;
        int result = StartPosition(numberOfMoves, moves);
        System.out.println("Number of times the ant reaches original position: " + result);
    }
    public static int StartPosition(int N, int[] A) {
        int currentPosition = 0;
        int count = 0;
        for (int i = 0; i < N; i++) {
            currentPosition += A[i];
            if (currentPosition == 0) {
                count++;
            }
        }
        return count;
    }
}

//Number of times the ant reaches original position: 2
```

```java
public class Arduino {

    public static int farthestCoordinate(int N, int[] A) {
        int currentPos = 0;
        int maxDistance = 0;

        for (int i = 0; i < N; i++) {
            currentPos += A[i];
            maxDistance = Math.max(maxDistance, Math.abs(currentPos));
        }

        return maxDistance;
    }

    public static void main(String[] args) {
        int N = 5;
        int[] A = {2, -1, 3, -4, 1};

        int result = farthestCoordinate(N, A);
        System.out.println("value :"+result);
    }
}

//value :4
```

```java
public class BirthdayParty {
    private static int maxPeopleWithEqualPieces(int N, int M) {
        // Calculate the greatest common divisor (GCD) of N and M
        int gcd = findGCD(N, M);
        // The maximum number of people is equal to the GCD
        return gcd;
    }
    private static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }
    public static void main(String[] args) {
        int N = 25, M = 10;
        int result = maxPeopleWithEqualPieces(N, M);
        System.out.println("Maximum number of people at the party: " + result);
    }
}

//Maximum number of people at the party: 5
```

```java
import java.util.Arrays;
class BoringArrays{
    public static void main(String[] args) {
        int [] a={10,5,1,8,4};
        int n=a.length;
        System.out.println(max(a,n));
    }
    public static int max(int [] a,int n){

        Arrays.sort(a);
        int maxscore=0;
        int i=n-2;

        while(i>0){
            maxscore+=Math.abs(a[i]-a[i+1]);
            i-=2;
        }
        return maxscore;
    }
}
//3
```

```java
import java.util.Arrays;

public class DebateProblem {

    public static int findDivisiblePair(int N, int K, int[] rollNumbers) {
        // Sort the roll numbers in ascending order
        Arrays.sort(rollNumbers);
        int startIndex = 0;
        int endIndex = N - 1;

        while (startIndex < endIndex) {
            // Calculate the sum of roll numbers for the current pair
            int currentSum = rollNumbers[startIndex] + rollNumbers[endIndex];

            // Check if the sum is divisible by K
            if (currentSum % K == 0) {
                return Math.min(rollNumbers[startIndex], rollNumbers[endIndex]);
            }

            // Move indices based on the comparison of currentSum with K
```

```
            if (currentSum < K) {
                startIndex++;
            } else {
                endIndex--;
            }
        }
        // If there is only one student left and twice their roll number is divisible by K
        if (2 * rollNumbers[startIndex] % K == 0) {
            return rollNumbers[startIndex];
        }

        // No valid pair found
        return -1;
    }
    public static void main(String[] args) {
        int N = 6;
        int K = 3;
        int[] rollNumbers = {5, 10, 2, 8, 15, 7};

        int result = findDivisiblePair(N, K, rollNumbers);
        System.out.println(result);
    }
}

//2
```

```
public class DiwaliContest {
    public static void main(String[] args) {
        int res = dc(8, 30);
        System.out.println(res);
    }
    static int dc(int np, int tm) {
        int tt = 240;
        int tl = tt - tm;
        int sum = 0;
        for (int i = 1; i <= np + 1; i++) {
            sum = sum + 5 * i;
            if (sum > tl) {
                return i - 1;
            }
        }
        return -1;
    }
}

//DiwaliContest 8
```

```java
public class EquationProblem {

    public static int findSetsCount(int N) {
        int count = 0;
        // Iterate over possible values of a, b, and c
        for (int a = 1; a <= N; a++) {
            for (int b = 1; b <= N; b++) {
                for (int c = 1; c <= N; c++) {
                    if (a*a + b*b + c*c + a*b + b*c + c*a == N) {
                        count++;
                    }
                }
            }
        }
        return count;
    }
    public static void main(String[] args) {
        int input_value = 6;
        int output_value = findSetsCount(input_value);
        System.out.println(output_value);
    }
}

//1
```

```java
public class FellisFunction {
    public static void main(String[] args) {
        int N = 3;
        int result = fellisFunction(N);
        System.out.println(result);
    }
    public static int fellisFunction(int N) {
        if (N == 0) {
            return 1;
        } else if (N == 1) {
            return 1;
        } else {
            return (fellisFunction(N - 1) + 7 * fellisFunction(N - 2) + (N / 4)) %
1000000007;
    }
}
}

//15
```

```java
public class FindOriginalInteger {

    public static int findOriginalInteger(int z1, int z2, int z3) {
        // Subtract 5 from each value to get back to the original values.
        int x1 = (z1 - 5) / 2;
        int x2 = (z2 - 5) / 2;
        int x3 = (z3 - 5) / 2;
        // Find the XOR of all three values.
        int xor = x1 ^ x2 ^ x3;

        // Find the bit that needs to be flipped.
        int flipBit = 0;
        for (int i = 0; i < 32; i++) {
            if (((xor >> i) & 1) == 1) {
                flipBit = i;
                break;
            }
        }
        // Flip the bit in x1 and return the result.
        return x1 ^ (1 << flipBit);
    }
    public static void main(String[] args) {
        int z1 = 15;
        int z2 = 20;
        int z3 = 25;
        int originalInteger = findOriginalInteger(z1, z2, z3);
        System.out.println("The original integer is: " + originalInteger);
    }
}

//The original integer is: 13
```

```java
public class FrogInPond {

    public static int maxPetalsRoute(int N, int[] A) {
        // Base cases
        if (N <= 2) {
            return 0;
        }
        int[] maxPetals = new int[N];
        // The frog can start from any lily pad, so initialize the first two values
        maxPetals[0] = A[0];
        maxPetals[1] = A[1];

        for (int i = 2; i < N; i++) {
```

```
            maxPetals[i] = Math.max(A[i] + maxPetals[i - 2], maxPetals[i - 1]);
        }
        return maxPetals[N - 1];
    }
    public static void main(String[] args) {
        // Example usage:
        int N = 5;
        int[] A = {2, 9, 3, 4, 5};
        int result = maxPetalsRoute(N, A);
        System.out.println(result);
    }
}

//14
```

```
import java.util.*;
class GeneratedNumbers{
    public static void main(String[] args) {
        System.out.println(performOpeartions(10, 2, 5));
    }
    public static int performOpeartions(int N,int A,int B){
        HashSet<Integer>hs=new HashSet<>();
        int n1=10;
        int n2=2;
        int n3=5,temp1=1,temp2=1;
        hs.add(n1);
        while(n1>0 && temp1>0 && temp2>0){
             temp1=n1-n2;
            temp2=n1-n3;
            if(temp1>0)
                hs.add(temp1);
            if(temp2>0)
                hs.add(temp2);
            if(temp1>temp2)
                n1=temp1;
            else
                n1=temp2;
        }

        int c=hs.size();
        return c;
    }
}

//8
```

```java
class IndexSorting {
    public static void main(String[] args) {
        int[] a = { 4, 5, 3, 2, 1 };
        int n = a.length;
        System.out.println(count(a, n));
    }
    public static int count(int[] a, int n) {
        int count = 0;
        for (int k = 1; k <= n; k++) {
            for (int i = 0; i < n; i++) {
                if (i != k - 1) {
                    for (int j = i + 1; j < n; j++) {
                        count++;
                        // break;
                    }
                }
            }
        }
        return count;
    }
}

// 16
```

```java
public class KeyboardPress {
    public static void main(String[] args) {
        String s="6004";
        System.out.println(minKeyPress(s));
    }
    public static int minKeyPress(String s) {
        int cno=1;
        int keyPress=-1;
        for(char c: s.toCharArray()){
            int digit=c-'0';
            if(digit==0){
                cno*=100;
            }
            else{
                cno=cno*10+digit;
            }
            keyPress++;
        }
        return keyPress;
    }
}

//3
```

```java
public class MagicString {
    public static void main(String[] args) {
        String s = "abcabc"; // Replace this string with the desired input
        int steps = findMinSteps(s);
        System.out.println("Minimum steps: " + steps);
    }

    public static int findMinSteps(String s) {
        int n = s.length();

        if (n == 0) {
            return 0; // An empty string is already a Magic String
        }

        char commonChar = s.charAt(0);
        int steps = 0;

        for (int i = 1; i < n; i++) {
            if (s.charAt(i) != commonChar) {
                steps++;
            }
        }
        return steps;
    }
}

//Minimum steps: 4
```

```java
public class MinArraySum {
    public static void main(String[] args) {
        int a[] = { 1, 4, 2, 9, 5 };
        System.out.println(toDebate(a));
    }
    public static int toDebate(int a[]){
        int c = 5;
        int b = 9;
        int avg = (b + c) / 2;
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            if (a[i] > avg) {
                sum += a[i];
            }
        }
        if(sum==0){
            return 0;
```

```
        }
        return sum;
    }
}

//9
```

```
public class NumberPyramid {

    public static void main(String[] args) {
        int row=5;
        int totalSum = 0;
        for (int i = 1; i <= row; i++) {
            for (int j = 1; j <= row - i; j++) {
                System.out.print(" ");
            }
            for (int j = i; j >= 1; j--) {
                System.out.print(j);
                totalSum += j;
            }
            for (int j = 2; j <= i; j++) {
                System.out.print(j);
                totalSum += j;
            }
            System.out.println();
        }
        System.out.println(totalSum);
    }
}

//     1
//    212
//   32123
//  4321234
// 543212345
// 65
```

```
import java.util.*;
public class OliviasGarden
{
    public static void main(String[] args)
    {
     int[] arr={2,2,5};
     int res1=og(arr,arr.length);
```

```
     System.out.println(res1);

    }
    static int og(int[] arr,int n)
    {
        Arrays.sort(arr);
        int sum=0;
        for(int i=0;i<arr.length;i++)
        {
           int res=0;
           res=arr[i]-arr[0];
           sum=sum+res;
        }
        return sum;
    }

}
//3
```

```
public class PairProblem {
    public static int fun(int[]a,int n){
        int c=0;
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                int r=a[i]*a[j];
                for(int k:a){
                    if(r==k){
                        c++;
                    }
                }
            }
        }
        return c;
    }
    public static void main(String[] args) {
        int[] a={1,2,3,4,5};
        int n=a.length;
        System.out.println(fun(a,n));
    }
}
//10
```

```
import java.util.Arrays;
public class PSBalance {
    public static void main(String[] args) {
        int[] a={1,2,3,4,5};
        int[] res=Balance(a);
```

```
        System.out.println(Arrays.toString(res));
    }
    public static int[] Balance(int[] a){
        int n=a.length;
        int[] r=new int[n];
        int sum=0;
        for(int i=0;i<n;i++){
            sum+=a[i];
            r[i]=Math.abs(sum-sumR(i+1,a));
        }
        return r;
    }
    public static int sumR(int j,int[] a){
        int n=a.length;
        int val=0;
        for(int i=j;i<n;i++){
            val+=a[i];
        }
        if(j==n){
            return 0;
        }
        return val;
    }
}
//[13, 9, 3, 5, 15]
```

```
public class ReduceTillZero {
    public static int fun(int x, int y) {
        if (y == 0) {
            return x;
        }
        if (x < y) {
            int t = x;
            x = y;
            y = t;
            fun(x, y);
        }
        else {
            int temp = x - y;
            x = y;
            y = temp;
            fun(x, y);
        }
        return x;
    }
    public static void main(String[] args) {
        int x = 8, y = 9;
        System.out.println(fun(x, y));
```

```
    }
}

//9
```

```
public class RoboRace {
    public static void main(String[] args) {
        int x = 6;
        int n = 3;
        int y = 7;
        int m = 4;
        int leastTime = findLeastTime(x, n, y, m);
        System.out.println("The least time at which Robotop and Robocop meet is: " +
leastTime + " seconds");
    }
    private static int findLeastTime(int x, int n, int y, int m) {

        while (x != y) {
            if (x < y) {
                x += n;
            } else {
                y += m;
            }
        }
        return x;
    }
}
//The least time at which Robotop and Robocop meet is: 15 seconds
```

```
    public class StringSwap {

    public static String performOperations(String s, int[] operations) {
        for (int op : operations) {
            if (op == 1) {
                s = swapFirstAndLast(s);
            } else if (op == 2) {
                s = swapFirstAndLastHalf(s);
            }
        }
        return s;
    }
    private static String swapFirstAndLast(String s) {
        char[] chars = s.toCharArray();
        char temp = chars[0];
        chars[0] = chars[chars.length - 1];
        chars[chars.length - 1] = temp;
```

```
        return new String(chars);
    }
    private static String swapFirstAndLastHalf(String s) {
        int length = s.length();
        int half = length / 2;
        char[] chars = s.toCharArray();
        for (int i = 0; i < half; i++) {
            char temp = chars[i];
            chars[i] = chars[i + half];
            chars[i + half] = temp;
        }
        return new String(chars);
    }
    public static void main(String[] args) {
        String inputString = "String";
        int[] operations = {1, 2, 1};
        String result = performOperations(inputString, operations);
        System.out.println("Final String: " + result);
    }
}

//Final String: rnSgti
```

```
public class TilingCost {

    public static int calculateTilingCost(int X, int Y) {
        if (X == 0 || Y == 0) {
            return 0;
        } else if (X == 1 || Y == 1) {
            return X * Y * ((int) Math.pow(2, X) + (int) Math.pow(3, Y) + 5);
        } else {
            return calculateTilingCost(X - 2, Y - 2) + ((2 * X) + (2 * Y) - 4) * ((int)
Math.pow(2, X) + (int) Math.pow(3, Y) + 5);
        }
    }
    public static void main(String[] args) {
        int input_X = 3;
        int input_Y = 5;
        int output_cost = calculateTilingCost(input_X, input_Y);
        System.out.println(output_cost);
    }
}

//3174
```

```java
public class WhatIsX {

    public static int findOriginalInteger(int z1, int z2, int z3) {
        // Subtract 5 from each value to get back to the original values.
        int x1 = (z1 - 5) / 2;
        int x2 = (z2 - 5) / 2;
        int x3 = (z3 - 5) / 2;

        // Find the XOR of all three values.
        int xor = x1 ^ x2 ^ x3;

        int flipBit = 0;
        for (int i = 0; i < 32; i++) {
            if (((xor >> i) & 1) == 1) {
                flipBit = i;
                break;
            }
        }
        // Flip the bit in x1 and return the result.
        return x1 ^ (1 << flipBit);
    }
    public static void main(String[] args) {
        int z1 = 15;
        int z2 = 20;
        int z3 = 25;
        int originalInteger = findOriginalInteger(z1, z2, z3);
        System.out.println("The original integer is: " + originalInteger);
    }
}

//The original integer is: 13
```