1. Simple & Segmented Sieve

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        boolean bool[]=new boolean[n+1];

        for(int i=2;i<n+1;i++){
            bool[i]=true;
        }

        for(int i=2;i*i<n+1;i++){
            if(bool[i]==true){
            for(int j=i*i;j<n+1;j=j+i){
                bool[j]=false;
            }
            }
        }

        for(int i=2;i<n+1;i++){
            if(bool[i]==true){
                System.out.print(i+" ");
            }
        }
    }
}
```

2. GCD(Euler Phi)

```java
import java.io.*;
import java.util.*;

public class Solution {

    static int gcd(int a,int b){
        while(b!=0){
            int temp=b;
            b=a%b;
            a=temp;
        }
        return a;
    }

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

       int n=x.nextInt();
       int sum=0;

       for(int i=1;i<n;i++){
           if(gcd(n,i)==1){
               sum+=i;
           }
       }

       System.out.println(sum);
    }
}
```

3. Strobogrammatic Number

```java
import java.util.*;

public class Solution {
    static List<String> findStrobogrammatic(int n) {
        return generate(n, n);
    }

    static List<String> generate(int n, int length) {
        if (n == 0) return Arrays.asList("");
        if (n == 1) return Arrays.asList("0", "1", "8");

        List<String> result = new ArrayList<>();
        for (String middle : generate(n - 2, length)) {
            if (n != length) result.add("0" + middle + "0");
            result.add("1" + middle + "1");
            result.add("8" + middle + "8");
            result.add("6" + middle + "9");
            result.add("9" + middle + "6");
        }
        return result;
    }

    public static void main(String[] args) {
        List<String> result = findStrobogrammatic(2);
        Collections.sort(result, Collections.reverseOrder()); // Sort in descending order
        System.out.println(String.join("  ", result)); // Use double spaces for output
    }
}

```

4. Remainder Theorem

```java
public class Solution {
     static int number(int num[],int rem[], int n){

          int temp=1;
          while(true){
              int i;
            for(i=0;i<n;i++){
               if(temp%num[i] != rem[i]){
                   break;
               }

            }
               if(n==i){
                   return temp;
               }
               temp++;
          }
      }

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

        int n=x.nextInt();

        int num[]=new int[n];
        int rem[]=new int[n];

        for(int i=0;i<n;i++){
            num[i]=x.nextInt();
        }
        for(int i=0;i<n;i++){
            rem[i]=x.nextInt();
        }

        int l=number(num,rem,n);

        System.out.println(l);

    }
}
```

5. Toggle the Switch

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        int n=x.nextInt();

        int res=(int) Math.sqrt(n);

        System.out.println(res);
    }
}
```

6. Alice Apple

```java
import java.util.Scanner;

public class Solution {

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

        int n=x.nextInt();

        int apple=0;
        int side=0;

        while(apple<n){
            side++;
            apple=8*side*side;
        }

        System.out.println(8*side);
    }
}

```

7. Binary Palindrome

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        int n=x.nextInt();

        String str=Integer.toBinaryString(n);
        StringBuilder rev=new StringBuilder(str);
        rev.reverse();
        if(str.equals(rev.toString())){
            System.out.println("yes");
        }
         else{
             System.out.println("no");
         }
    }
}
```

8. **Karatsuba**

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        String a=x.next();
        String b=x.next();

        int a1=Integer.parseInt(a,2);
        int a2=Integer.parseInt(b,2);

        System.out.println(a1*a2);
    }
}
```

9. Longest Sequence of 1 after flipping a bit

```java
import java.io.*;
import java.util.*;

public class Solution {

    static int flipBit(int n){
        int prev=0,max=0,curr=0;

        while(n!=0){
            if(n%2==1){
                curr++;
            }
            else{
                prev=curr;
                curr=0;
            }
            max=Math.max(max,prev+curr+1);
            n>>>=1;
        }
        return max;
    }

    public static void main(String[] args) {
       Scanner x=new Scanner(System.in);
        int n=x.nextInt();
        int l=flipBit(n);
        System.out.println(l);
    }
}
```

10. Nibble Swap

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String arg[]){
        Scanner y=new Scanner(System.in);
        int x = y.nextInt();

        int a = x % 16;
        int b = x / 16;
        int ans = a * 16 + b;

        System.out.println(ans);
    }
}
```

11. Selection and Quick Sort

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner x=new Scanner(System.in);
        int n=x.nextInt();
        int arr[]=new int[n];

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

        for(int i=0;i<n-1;i++){
            int mini=i;
            for(int j=i+1;j<n;j++){
                if(arr[j]<arr[mini]){
                    mini=j;
                }
            }
            int temp=arr[i];
            arr[i]=arr[mini];
            arr[mini]=temp;
        }

        for(int i=0;i<n;i++){
            System.out.print(arr[i]+" ");
        }
    }
}
```

12. Armstrong Number

```java
import java.util.*;

class Main {
    public static void main(String args[]){
        Scanner x=new Scanner(System.in);

        int n=x.nextInt();
        int y=n;

        int sum=0;

        while(n>0){
            int i=n%10;
            sum+=(i*i*i);
            n=n/10;
        }

        if(sum==y){
            System.out.println("true");
        }
        else{
            System.out.println("false");
        }
    }
}
```

13. Palindrome

```java
import java.util.*;

class Main {
    public static void main(String args[]){
        Scanner x=new Scanner(System.in);

        int n=x.nextInt();
        int y=n;

        int sum=0;

        while(n>0){
            int i=n%10;
            sum=(sum*10)+i;
            n=n/10;
        }

        if(sum==y){
            System.out.println("true");
        }
        else{
            System.out.println("false");
        }
    }
}
```

14. Maneuvering a Cave Problem (Possible paths from top left to bottom right of a matrix)

```java
import java.io.*;
import java.util.*;

public class Solution{

    static int someFunc(int m,int n){
        if(m==1 || n==1){
            return 1;
        }
        return someFunc(m-1,n)+someFunc(m,n-1);
    }

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

        int m=x.nextInt();
        int n=x.nextInt();

        System.out.println(someFunc(m,n));
    }
}
```

15. Josepheus Trap

There are n people standing in a circle waiting to be executed. The counting begins at some point in the circle and proceeds around the circle in a fixed direction. In each step, a certain number of people are skipped and the next person is executed. The elimination proceeds around the circle (which is becoming smaller and smaller as the executed people are removed), until only the last person remains, who is given freedom. Given the total number of persons n and a number k which indicates that k-1 persons are skipped and kth person is killed in a circle. The task is to choose the place in the initial circle so that you will be the only remaining person alive.

```java
import java.io.*;
import java.util.*;

public class Solution{

    static int trap(int n,int k){
        if(n==1){
            return 1;
        }
        else{
            return (trap(n-1,k)+(k-1))%n + 1;
        }
    }
    public static void main(String args[]){
        Scanner x=new Scanner(System.in);

        int n=x.nextInt();
        int k=x.nextInt();

        System.out.println(trap(n,k));
    }
}
```