1.

```java
// Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti, hogy prímszám-e, vagy sem!
public static boolean primE(int n) {
    if (n <= 1) return false;
    for (int i = 2; i <= Math.sqrt( a:n); i++) {
        if (n % i == 0) return false;
    }
    return true;
}
```

2.

```java
// Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti,
// hogy tökéletes szám-e, vagy sem! (pozitív osztóinak összege a szám kétszerese)
public static boolean tokeletesSzamE(int n) {
    int osszeg = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) osszeg += i;
    }
    return osszeg == (n * 2);
}
```

3. import java.util.Random;

```java
// Írjon olyan függvényt vagy metódust, amely egy karakterláncban vagy sztringben
// véletlenszerűen összekeveri a karaktereket (véletlenszám-generátor használható)!
public static String kever(String s) {
    char[] betuk = s.toCharArray();
    Random random = new Random();
    char temp;

    for (int i = 0; i < betuk.length; i++) {
        int j = random.nextInt( bound:betuk.length);
        temp = betuk[i];
        betuk[i] = betuk[j];
        betuk[j] = temp;
    }

    return new String( chars:betuk);
}
```

4.

```java
// A következő közelítő formulát használva írjon függvényt vagy metódust, amely egy valós
// szám négyzetgyökét adja vissza! Használja az xk+1=1/2*(xk+a/xk) sorozatot, amely a
// négyzetgyökéhez konvergál, ha x1=1.
public static double negyzetgyok(double a) {
    double x = 1;
    for (int i = 0; i < 10; i++) {
        x = 0.5 * (x + a / x);
    }
    return x;
}
```

5.

```java
// Írjon függvényt vagy metódust, amely egy valós szám köbgyökét adja vissza!
// Használja az xk+1=1/3*(2*xk+a/xk2) sorozatot, amely 'a' köbgyökéhez konvergál, ha x1=1.
public static double kobgyok(double a) {
    double x = 1;
    for (int i = 0; i < 100; i++) {
        x = (2 * x + a / (x * x) ) / 3;
    }
    return x;
}
```

6.

```java
// Írjon függvényt vagy metódust, amely kiszámolja az n-edik Fibonacci számot!
// A Fibonacci sorozatot az an=an-2+an-1 rekurzióval definiálja (n>2), ahol a1=a2=1.
public static int fibonaccsi(int n) {
    if (n < 2) return 1;
    else return fibonaccsi(n-2) + fibonaccsi(n-1);
}
```

7.

```java
// Írjon olyan függvényt vagy metódust, amely egy természetes számhoz visszaadja azt a
// legnagyobb egész kitevős hatványát, amely még éppen kisebb, mint 567!
public static int hatvany(int n) {
    if (n <= 1) return 1;
    int szam = n;
    while (szam * n < 567) {
        szam *= n;
    }
    return szam;
}
```

8.

```java
// Írjon olyan függvényt vagy metódust, amely egy természetes szám esetén kiírja,
// hogy a 9-es számjegyből hány darabot tartalmaz (ne alakítsa át sztringgé/karaktertömbbé)!
public static int hanyKilences(int n) {
    int darab = 0;
    while (n > 0) {
        if (n % 10 == 9) {
            darab++;
        }
        n = n / 10;
    }
    return darab;
}
```

9.

```java
// Írjon olyan függvényt vagy metódust, amely egy természetes számról eldönti, hogy a kettes
// számrendszerbeli felírásában a jobbról második bitje 1 vagy 0 (ne alakítsa át sztringgé/karaktertömbbé)!
public static int jobbrolMasodikBit(int n) {
    n /= 2;
    return n % 2;
}
```

10.

```java
// Írjon olyan függvényt vagy metódust, amelynek paramétere egy 1 < x < 10 természetes szám,
// és kiírja az 1,3,4,6,7,9,10,12,... sorozat első öt x-szel osztható elemét, azaz a sorozat
// i+1-edik tagja 2-vel nagyobb az i-ediknél, ha i páratlan, s eggyel nagyobb az i-ediknél, ha i páros!
public static void sorozat(int x) {
    int db = 0;
    int i = 1;
    int sorozatN = 1;
    while (db < 5) {
        if (sorozatN % x == 0) {
            System.out.println( x:sorozatN);
            db++;
        }

        if (i % 2 == 1) {
            sorozatN += 2;
        } else {
            sorozatN++;
        }
        i++;
    }
}
```

11.

```java
// Írjon olyan függvényt vagy metódust, amely a paraméterében megadott természetes szám
// pozitív osztóinak számával tér vissza!
public static int osztokSzama(int n) {
    int db = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            db++;
        }
    }
    return db;
}
```

12.

```java
// Írjon olyan függvényt vagy metódust, amely egy karakterláncból vagy sztringből a
// számjegyek kivételével minden karaktert eltávolít!
public static String eltavolit(String s) {
    String eredmeny = "";
    for (int i = 0; i < s.length(); i++) {
        char c = s.charAt( index:i);
        if (c >= '0' && c <= '9' ) {
            eredmeny += c;
        }
    }
    return eredmeny;
}
```

13.

```java
// Írjon olyan függvényt vagy metódust, amely egy karakterláncról vagy sztringről eldönti,
// hogy palindróma-e! (Balról olvasva ugyanaz, mint jobbról olvasva.)
public static boolean palindroma(String s) {
    for (int i = 0; i <= s.length()/2; i++) {
        if (s.charAt( index:i) != s.charAt(s.length()-1-i)) {
            return false;
        }
    }
    return true;
}
```

14.

```java
// Írjon olyan függvényt vagy metódust, amely egy, az angol ábécé betűit tartalmazó
// karakterláncban vagy sztringben minden szó kezdőbetűjét nagybetűre alakítja!
public static String szavakNagyKezdobetu(String mondat) {
    String[] szavak = mondat.split( regex:"\\s");
    String eredmeny = "";
    for (String s : szavak) {
        eredmeny += s.substring( beginIndex:0,  endIndex:1).toUpperCase() + s.substring( beginIndex:1) + " ";
    }
    return eredmeny;
}
```

15.

```java
// Írjon olyan függvényt vagy metódust, amely egy karakterláncból vagy sztringből eltávolítja
// egy megadott karakter összes előfordulását!
public static String karatkerekEltavolit(String s, char c) {
    String eredmeny = "";
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt( index:i) != c) {
            eredmeny += s.charAt( index:i);
        }
    }
    return eredmeny;
}
```

16.

```java
// Írjon olyan függvényt vagy metódust, amely megszámolja egy adott karakterlánc vagy
// sztring összes előfordulását egy másik karakterláncban vagy sztringben!
public static int sztringElofordulasa(String s, String keres) {
    int db = 0;
    int index = 0;
    while ( (index = s.indexOf( str:keres,  fromIndex:index)) != -1 ) { // 2. paraméter, hogy melyik indextől kezdje a keresést
        db++;
        index += keres.length();
    }
    return db;
}
```

17.

```java
// Írjon olyan függvényt vagy metódust, amely kiírja az angol kisbetűs ábécé azon betűit, melyek ASCII kódja négyzetszám!
public static void negyzetASCII() {
    for (int i = 'a'; i <= 'z'; i++) {
        if (Math.sqrt( a:i) % 1 == 0) {
            System.out.println((char) i);
        }
    }
}
```

18. import java.util.Random;

```java
// Írjon olyan függvényt vagy metódust, amely előállít egy 5 karakterből (angol kisbetűs ábécé
// karaktereit használva) álló véletlen karakterláncot vagy sztringet! Biztosítsa, hogy minden 5
// hosszú különböző betűkből álló sztring egyenlő valószínűséggel kerüljön kiválasztásra,
// feltéve, hogy a választott programozási nyelv véletlenszám-generátora egyenletes eloszlást biztosít!
public static String randomString() {
    Random r = new Random();
    String s = "";
    char c;
    int i;
    while (s.length() < 5) {
        c = (char)(r.nextInt('z' - 'a' + 1) + 'a');
        for (i = 0; i < s.length(); i++) {
            if (s.charAt( index: i) == c) break;
        }
        if (i == s.length()) s += c;
    }
    return s;
}
```

19. import java.util.Random;

```java
// Írjon olyan függvényt vagy metódust, amely egy karakterláncba vagy sztringbe beszúr
// egy „a” karaktert véletlenül választott pozícióba (véletlenszám-generátor használható)!
public static String karaktertBeszur(String s) {
    Random random = new Random();
    int index = random.nextInt(s.length() + 1); // + 1, mivel a végére is szúrhatja
    return s.substring( beginIndex: 0,  endIndex: index) + 'a' + s.substring( beginIndex: index);
}
```

20.

```java
// Adjon olyan függvényt vagy metódust, ami adott két pozitív egész paramétere esetén
// megadja (n alatt a k)=n!/k!(n-k)! értékét. Használjon rekurziót!
public static int kombinacio(int n, int k) {
    if (k > n) return 0;
    if (k == 0 || k == n) return 1;
    return kombinacio(n-1, k-1) + kombinacio(n-1, k);
}
```

21.

```java
// Adjon olyan metódust vagy függvényt, ami eldönti, hogy a paramétereként megadott
// (pozitív egészekből álló) nemüres tömbben van-e olyan szám, ami az összes többit osztja.
// (Maradékszámító függvény használható).
public static boolean vanEMindentOszto(int[] szamok) {
    int db;
    for (int i = 0; i < szamok.length; i++) {
        db = 0;
        for (int j = 0; j < szamok.length; j++) {
            if (szamok[j] % szamok[i] == 0) db++;
            if (db == szamok.length) return true;
        }
    }
    return false;
}
```

22.

```java
// Adjon olyan metódust vagy függvényt, ami eldönti, hogy a paramétereként megadott
// (pozitív egészekből álló) nemüres tömbben van-e olyan szám, ami az összes többinél többször fordul elő.
public static void tobbszorMintATobbi(int[] szamok) {
    int maxIndex = 0;
    for (int i = 1; i < szamok.length; i++) {
        if (szamok[i] > szamok[maxIndex]) {
            maxIndex = i;
        }
    }

    int[] gyakorisagok  = new int[szamok[maxIndex]+1];

    for (int i = 0; i < szamok.length; i++) {
        gyakorisagok[szamok[i]]++;
    }

    for (int i = 0; i < szamok.length; i++) {
        System.out.println(szamok[i]+" "+gyakorisagok[szamok[i]]);
    }
}
```

23.

Adjon olyan metódust vagy függvényt, ami visszaadja, hogy a paramétereként megadott (pozitív egészekből álló) nemüres tömbben melyik index az, ahol a leghosszabb folyamatosan növekvő részsorozat kezdődik. Ha több ilyen index is van, az utolsót adja vissza

24

```java
// Adjon olyan metódust vagy függvényt, ami visszaadja, hogy a paramétereként megadott
// (pozitív egészekből álló) nemüres tömbben melyik az a legkisebb index, amire az index
// előtti elemek összege meghaladja a tömb első két elemének szorzatát. Ha nincs ilyen, 0-t adjon vissza.
public static int zv24(int[] szamok) {
    int szorzat = 0;
    int osszeg = 0;
    if (szamok.length > 1) szorzat = szamok[0]*szamok[1];
    for (int i = 0; i < szamok.length; i++) {
        if (szorzat < osszeg) return i;
        osszeg += szamok[i];
    }
    return 0;
}
```

25.

```java
// Adjon egy metódust vagy függvényt, ami paraméterként adott s sztring/karaktertömb, c
// karakter és n pozitív egész szám esetén megadja, hogy a c karakter n-edik előfordulása
// hányadik pozíción van az „s” sztringben.
public static int pozicio(String s, char c, int n) {
    int db = 0;
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt( index:i) == c) {
            db++;
            if (n == db) return i;
        }
    }
    return -1;
}
```

26.

```java
// Adjon metódust vagy függvényt, ami a paraméterként kapott, egészekből álló rendezett
// tömbben a tömb hosszának logaritmusával arányos lépésszám alatt megkeresi a
// paraméterként kapott egész első előfordulásának indexét, illetve ha nincs ilyen, akkor -1-et ad vissza. (pl. a bináris keresés)
public static int binaris(int[] tomb, int n) {
    int also = 0;
    int felso = tomb.length-1;

    while (also <= felso) {
        int kozepso = (also+felso)/2;
        if (tomb[kozepso] == n) return kozepso;
        else if (n < tomb[kozepso]) felso = kozepso-1;
        else if (n > tomb[kozepso]) also = kozepso+1;
    }
    return -1;
}
```

27.

```java
// Írjon függvényt vagy metódust, mely visszaadja két egész paramétere szorzatának balról
// második számjegyét! (a megoldás során ne használjon sztringeket/karaktertömböket)
public static int balrolMasodik(int a, int b) {
    int szorzat = Math.abs(a*b);
    while (szorzat > 100) {
        szorzat /= 10;
    }
    return szorzat % 10;
}
```

28.

```java
public static boolean atloban2(char[][] c) {
    for (int i = 0; i < 5; i++) {
        for ( int j = 0; j < 5; j++) {
            for ( int k = 0; k < 5; k++) {
                if ( c[j][k] == c[i][i] && j!=k ) return true;
            }
        }
    }
    return false;
}
```

29.

```java
// Írjon függvényt vagy metódust, mely valós típusú paraméterének azt a számjegyét adja
// vissza, amelyik a tizedes pont után áll! (a megoldás során ne használjon sztringeket/karaktertömböket)
public static int tized(double d) {
    d = Math.abs( a:d);
    return (int) (d * 10) % 10;
}
```

30.

```java
// Írjon függvényt vagy metódust, mely pozitív egész paraméterét fordítva adja vissza, pl.
// fordit(234) eredménye 432! (a megoldás során ne használjon sztringeket/karaktertömböket)
public static int fordit(int n) {
    int n2 = 0;
    while (n != 0) {
        n2 = n2 * 10 + (n % 10);
        n /= 10;
    }
    return n2;
}
```

31.

Írjon függvényt vagy metódust, mely a paraméterként kapott 10x10-es mátrixról eldönti, hogy van-e olyan eleme, mely sorában nagyobb és oszlopában pedig kisebb a többi elemnél!

32.

```
// Írjon függvényt vagy metódust, mely visszaadja, hogy k-tól m-ig hány olyan szám van,
// melyeknek n db valódi osztója van! (n, k és m paraméter).
public static int valodiOsztok(int k, int m, int n) {
    int db = 0;
    int osztok = 0;
    for (int i = k; i <= m; i++) {

        for (int j = 2; j < i; j++) {
            if (i % j == 0) osztok++;
        }
        if (osztok == n) db++;
        osztok = 0;
    }
    return db;
}
```

33.

```
// Írjon függvényt vagy metódust, mely visszaadja, hogy két pozitív egész paraméterének
// legkisebb közös többszöröse hány számjegyből áll kettes számrendszerben.
public static int lkktHanyBinarisSzamjegy(int a, int b) {
    int max = Math.max(a, b);
    int lkkt = max;
    while (lkkt % a != 0 || lkkt % b != 0) {
        lkkt += max;
    }

    int szamjegy = 0;
    while (lkkt > 0) {
        szamjegy++;
        lkkt /= 2;
    }
    return szamjegy;
}
```