```
import java.util.Random;

public class Main {

   //1. Eldonti egy számról, prím szám-e.
   public static boolean PrimE(int n){
      if(n<=1) return false;
      for(int i = 2; i <= n/2; i++) if(n%i == 0) return false;
      return true;
   }

   //2. Eldonti egy számról, tökéletes szám-e! (Pozitív osztóinak összege n*2)
   public static boolean TokeletesE(int n){
      int sum = 0;
      if(n<1) return false;
      for(int i = 1; i <= n; i++) if(n%i == 0) sum+=i;
      return sum/2 == n;
   }

   //3. Véletlenszerűen összekeveri a karaktereket a karakterláncban.
   public static char[] Kever(char[] chars){
      Random r = new Random();
      int a;
      char temp;
      for(int i = 0; i < chars.length; i++){
         temp = chars[i];
         a = r.nextInt(chars.length);
         chars[i] = chars[a];
         chars[a] = temp;
      }
      return chars;
   }

   //4. Egy valós szám négyzetgyökét adja vissza!
   public static double Negyzetgyok(double d){
      if(d < 0) return -1;
      double gyok = 1;
      for(int i = 0; i < 50; i++){
         gyok = (1.0/2.0)*(gyok+(d/gyok));
      }
      return gyok;
   }
```

```
//5. Egy valós szám köbgyökét adja vissza!
  public static double Kobgyok(double d){
    double gyok = 1;
    for(int i = 0; i < 50; i++){
      gyok = (1.0/3.0)*(2*gyok+(d/Math.pow(gyok,2)));
    }
    return gyok;
  }
  //6. Visszaadja az n-edik fibonacci számot!
  public static int Fibonacci(int n){
    if(n < 2)
      return n;
    return Fibonacci(n-2) + Fibonacci(n-1);
  }

  //7.Írjon olyan függvényt vagy metódust, amely egy természetes számhoz visszaadja azt a
       legnagyobb egész kitevős hatványát, amely még éppen kisebb, mint 567!
  private static int hatvany(int n) {
    if(n <= 1) return 1;
    int n0 = n;
    while(n*n0<567){
      n *= n0;
    }
    return n;
  }
  //8. Természetes szám esetén kiírja, hogy a 9-es számjegyből hány darabot tartalmaz
  public static void Kilencesek(int n){
    int db = 0;
    while(n>0){
      if(n%10 == 9) db++;
      n /= 10;
    }
    System.out.println(db);
  }
```

```
 //9. Eldönti, hogy kettes számrendszerbeli felírásban jobbról második bitje 1
  //vagy 0
  public static int JobbrolMasodik(int n){
    n /= 2;
    return n % 2;
  }

  //10. Olyan függvény, aminek paramétere 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 Sorozatos(int x){
   int db = 0;
   int sorozatN = 1;
   int i = 1;
   while(db<5){
      if(sorozatN % x == 0){
         System.out.println(sorozatN);
         db++;
      }
      if(i % 2 == 0){
         sorozatN++;
      }else{
         sorozatN += 2;
      }
      i++;
   }
}

//11. Paraméterben megadott természetes szám pozitív osztóinak számával tér vissza
public static int PozitivOsztokSzama(int n){
   int db = 0;
   for(int i = 1; i <= n; i++){
      if(n%i == 0) db++;
   }
   return db;
}
```

```
//12. Stringből számjegyek kivételével minden karaktert eltávolít
   public static String KarakterEltavolit(String s){
      StringBuilder sb = new StringBuilder(s);
      int hossz = s.length();
      for(int i = 0; i < hossz; i++){
         if(sb.charAt(i) < '0' || sb.charAt(i) > '9'){
            sb.deleteCharAt(i);
            hossz--;
```

```
            i--;
        }
    }
    return sb.toString();
}

//13.Stringtől eldönti, palindróma-e
public static boolean Palindroma(String s){
    for(int i = 0; i < s.length(); i++){
        if(s.charAt(i) != s.charAt(s.length()-1-i)) return false;
    }
    return true;
}

//14. Angol abc betűit tartalmazó stringben minden szó kezdőbetűjét nagyra
//alakítja
public static char[] NagyKezdo(char[] chars){
    for(int i = 0; i < chars.length; i++){
        if(i == 0 && chars[i] != ' ' && 'a' < chars[i] && chars[i] < 'z') chars[i] = (char) (chars[i]-32);
        if(i != 0 && chars[i-1] == ' '&& 'a' < chars[i] && chars[i] < 'z') chars[i] = (char) (chars[i]-32);
    }
    return chars;
}
//15. Eltávolít összes megadott előfordulást
public static String EltavolitKaraktert(String s, char c){
    StringBuilder sb = new StringBuilder(s);
    int hossz = sb.length();
    for(int i = 0; i < hossz; i++){
        if(sb.charAt(i) == c){
            sb.deleteCharAt(i);
            hossz--;
            i--;
        }
    }
    return sb.toString();
}



//16. Megszámolja egy karakterlánc vagy string összes előfordulását
    //egy másik stringben
    public static int MegszamolElofordulast(String s1, String s2){
        int db = 0, j;
        for(int i = 0; i < s1.length()-s2.length(); i++){
            for(j = 0; j < s2.length(); j++){
                if(s1.charAt(i+j) != s2.charAt(j)) break;
            }
            if(j == s2.length())
```

```
            db++;
        }
        return db;
    }

    //17. Kiírja az ASCII kisbetűket, amiknek kódja négyzetszám
    public static void KiirNegyzetASCII(){
        for(int i = 'a'; i <= 'z'; i++){
            if(Math.sqrt(i) % 1 == 0) System.out.println((char) i);
        }
    }

    public static void main(String[] args) {
        KiirNegyzetASCII();
    }

    //18. 5 hosszú különböző betűkből álló Stringet ad vissza
    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(i) == c) break;
            }
            if (i == s.length()) s += c;
        }
        return s;
    }

    //19. Beszúr 1 'a' karaktert 1 random helre
    private static String beszur(String s) {
        StringBuilder sb = new StringBuilder(s);
        Random r = new Random();
        sb.insert(r.nextInt(sb.length()+1),'a');
        return sb.toString();
    }

    //20.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 fgv(int n, int k){
        if(k>n) return 0;
        if(k == 0 || k == n) return 1;
        return fgv(n-1,k-1) + fgv(n-1,k);
    }
}
```

```
//21. 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.
   public static boolean osztojaE(int[] szamok){
      int n;
      for(int i=0; i< szamok.length; i++){
         n = 0;
         for(int j=0;j< szamok.length; j++){
            if(szamok[j]%szamok[i] == 0) n++;
            if(n==szamok.length) return true;
         }
      }
      return false;
   }

//22. 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.
private static boolean tobbszorMintTobbi(int[] t) {
      int legN = -1;
      for(int i = 0; i < t.length; i++) if(legN < t[i]) legN = t[i];
      int[] tomb = new int[legN];
      for(int i = 0; i < t.length; i++) tomb[t[i]-1]++;
      for(int i = 0; i < tomb.length-1; i++){
         for(int j = 0; j < tomb.length-1-i; j++){
            if(tomb[j] < tomb[j+1]){
               int temp = tomb[j];
               tomb[j] = tomb[j+1];
               tomb[j+1] = temp;
            }
         }
      }
      if(tomb.length > 1 && tomb[0] > tomb[1] && tomb[1] != 0) return true;
      return false;
   }
//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.
public static int resz(int[] szamok){
      int hossz = 0;
      int ind = 0;
      int leg = 0;
      int i = 0;
```

```
    for(i = 0; i < szamok.length-1; i++){
      if(szamok[i] < szamok[i+1]){
        hossz++;
      }else{
        if(leg <= hossz){
          leg = hossz;
          ind = i - hossz;
        }
        hossz = 0;
      }
    }
    if(leg <= hossz){
      ind = i - hossz;
    }
    return ind;
  }
```

//24. 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 fgv(int[] szamok){
    int szorzat = 0;
    int osszeg = 0;
    if(1 < szamok.length) szorzat = szamok[0] * szamok[1];
    for(int i = 0; i < szamok.length; i++){
      if(szorzat < osszeg) return i;
      osszeg += szamok[i];
    }
    return 0;
  }
```

//25. 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 Hanadik(String s, char c, int n){
    for(int i = 0; i < s.length(); i++){
      if(s.charAt(i) == c) n--;
      if(n == 0) return i;
    }
    return -1;
  }
```

//26. 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[] t,int n){
    int also = 0, felso = t.length-1;
    while(also <= felso){
      int kozepso = (also+felso)/2;
      if(t[kozepso] == n) return kozepso;
      else if(n < t[kozepso]) felso = kozepso-1;
      else if(t[kozepso] < n) also = kozepso+1;
    }
    return -1;
  }
```

//27. 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 n, int m){
    int a = Math.abs(n*m);
    int n1 = 0,n2 = 0;
    while(a!=0){
      n2 = n1;
      n1 = a%10;
      a = a/10;
    }
    return n2;
  }
```

//28. Írjon függvényt vagy metódust, mely eldönti, hogy a paraméterként kapott 5x5-ös /karakterekből álló/ tömbben a főátlóban van-e olyan elem, mely a főátlón kívül is megjelenik a tömbben!

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

//29. 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 tizedesUtan(double d){
   d = Math.abs(d);
   return (int)((d-(int)d)*10);
}
```

//30. 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 = 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!

```
public static boolean matrixdont2(int[][] matrix){
    boolean sorLeg, oszlopLeg;
    for(int i = 0; i < 10; i++){
       for(int j = 0; j < 10; j++){
          sorLeg = true;
          oszlopLeg = true;
          for(int k = 0; k < 10; k++){
             if(matrix[i][j] <= matrix[i][k] && !(j==k)) sorLeg = false;
          }
          for(int l = 0; l < 10; l++){
             if(matrix[i][j] >= matrix[l][j] && !(i==l)) oszlopLeg = false;
```

```
            }
            if(sorLeg && oszlopLeg) return true;
        }
    }
    return false;
}
```

//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 n, int k, int m){
    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 binarisban(int a, int b){
    int n = 0;
    for(int i = a; i > 0; i--){
        if(a%i == 0 && b%i == 0){
            n = (a*b)/i;
            break;
        }
    }
    return (int)Math.ceil(Math.log(n+1)/Math.log(2));
}
```