Izzivi (četrtič)

(1) Vsaka od logičnih spremenljivk `a1`, `a2`, ..., `a9` ima z verjetnostjo $\frac{1}{2}$ vrednost `true`. Kolikšna je verjetnost, da ima sledeči izraz tudi vrednost `true`?

```
a1 ? a2 : a3 ? a4 : a5 ? a6 : a7 ? a8 : a9
```

Pogojni operator je desno asociativen: izraz `a ? b : c ? d : e` se izračuna kot `a ? b : (c ? d : e)`.

(2) Kaj izpiše sledeči program?

```
public class Program {
    public static void main(String[] args) {
        int a = 3;
        int b = 4;
        zamenjaj(a, b);
        System.out.println(a);
        System.out.println(b);
    }

    public static void zamenjaj(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
    }
}
```

(3) Tipa `float` in `double` nam smrdita. Od tod sledi, da nam smrdi tudi metoda `Math.sqrt`, ki smo jo uporabili za določanje največjega delitelja, s katerim je še smiselno preverjati kandidata za praštevilo. Napišite metodo

```
public static long koren(long stevilo)
```

ki vrne navzdol zaokroženo vrednost korena podanega števila. Lahko predpostavite, da parameter `stevilo` pripada intervalu $[1, 10^{18}]$. Metoda mora tudi za največje možne vrednosti parametra rezultat vrniti v delčku sekunde.

(4) Prepišite sledečo metodo tako, da ne bo uporabljala rekurzije:

```
public static void izpisi(int n) {
    if (n > 0) {
        System.out.print("A");
        izpisi(n - 1);
        System.out.print("B");
    }
}
```

(5) Napišite metodo

`public static void izpisiStevke(int n)`

ki po vrsti izpiše števke števila `n` (vsako števko naj izpiše v svojo vrstico). Metoda ne sme uporabljati zank, od podatkovnih tipov pa lahko uporablja zgolj tipa `int` in `boolean`. Metoda lahko kliče pomožne metode, vendar pa morajo tudi te spoštovati ista pravila.

(6) »Število je sodo, če je njegov predhodnik lih, in liho, če je njegov predhodnik sod,« v družbi svojega najzvestejšega asistenta modruje profesor Doberšek. »To pomeni, da lahko metodi za preverjanje sodosti oziroma lihosti napišem enostavno takole:«

```
public static boolean jeSodo(int n) {
    return jeLiho(n - 1);
}

public static boolean jeLiho(int n) {
    return jeSodo(n - 1);
}
```

»To bi moralo delovati, mar ne, Jože?«

»Načeloma da,« v slogu znamenitih šal z radia Erevan prične asistent Slapšak, praskajoč se po malomarno obriti bradi, »a se mi kljub temu zdi, da nekaj manjka.«

»Kaj naj bi to bilo?«

»Res ne bi vedel. Bi povprašal Genovefo, a ravno predava na temo ... hm, kako je že rekla? ... robnih pogojev, da, robnih pogojev!«

»Robnih pogojev! Saj vem, da rada hodi po robu, a da bi o svojih podvigih še predavala?! Le kam smo prišli! *O tempora, o mores!*«

Dopolnite metodi tako, da bosta vrnili `true` natanko tedaj, ko bo podano število sodo oziroma liho. Vaše dopolnitve si seveda ne smejo pomagati z operatorjem `%`, lahko pa predpostavite, da je parameter `n` v obeh funkcijah pozitivno število.

(7) Podani sta sledeči metodi:

```
public static int g() {
    int stevec = 1;
    boolean b = true;
    while (b = f(b)) {  // pozor: =, ne ==
        stevec++;
    }
    return stevec;
}

public static boolean f(boolean b) {
    return (b) ? (Math.random() < 0.8) : (true);
}
```

Če je $0 \leqslant k \leqslant 1$, potem ima izraz `(Math.random() < k)` z verjetnostjo k vrednost `true`.

Recimo, da metodo `g` n-krat pokličemo in z a_n označimo povprečje njenih rezultatov. Izračunajte $\lim_{n\to\infty} a_n$.