TÖL101G - Tölvunarfræði 1
Vikublað 2
Lausn

Æfingar

1.2.20

```
public class Dice {
    public static void main(String[] args) {
        int roll1 = 1 + (int) (6 * Math.random());
        int roll2 = 1 + (int) (6 * Math.random());

        System.out.println((roll1+roll2));
    }
}
```

1.3.6

1. `for (i = 0, j =0; i < 10; i++) j += i;`

   Gildið á j verður 45, enda reiknar þetta út summuna $0 + 1 + \ldots + 9 = 45$.

2. `for (i = 0, j =1; i < 10; i++) j += j;`

   Gildið á j verður 1024 því `j += j` er jafngilt `j = 2*j`. Því verður gildið á j $1, 2, 4, 8, 16, 32, 62, 128, 256, 512$ og lokst 1024.

3. `for (j = 0; j < 10; j++) j += j;`

   Hér er j tvöfaldað inni í for lykkjunni og í lok hverrar lykkju er gildið hækkað um 1. Gildið á j verður því $0, 0, 1, 2, 3, 6, 7, 14, 15$ í hvert skipti sem það breytist. Lokagildið er því 15.

4. `for (i = 0, j =0; i < 10; i++) j += j++;`

   Hér verður að athuga hvað `j += j++` þýðir nákvæmlega. Segðin `j++` hækkar gildið á `j` um 1 og skilar gamla gildinu sem niðurstöðu. `j += j++` er jafngilt því að skrifa `j = j + (j++)`. Þegar þessi setning er framkvæmd er fyrst reiknað upp úr hægri hliðinni og loks er sú niðurstaða geymd í breytunni `j`. Þegar `j` er 0 þá er reiknað upp úr `j + (j++)` sem skilar $0 + 0 = 0$, og um leið og reiknað er upp úr `j++` er gildið á `j` hækkað um 1. Þá hefur `j` tímabundið gildið 1, en þegar búið er að reikna upp úr hægri hliðinni fæst niðurstaðan 0 sem er geymd í breytunni `j` og skrifar því yfir "hækkunin" sem að var framkvæmd fyrst. Því verður gildið á `j` alltaf 0. Ef þessi útskýring er óskiljanleg er ágætt að keyra þetta dæmi í jeliot forritinu.

1.3.16

a. `for (int i = 1; i <= N; i++) sum += 1 / (i*i);`

Þetta skilar ekki réttri útkomu. Í fyrsta lagi er `1/(i*i)` heiltöludeiling og verður því 0 nema þegar `i == 1`. Auk þess þegar `i == 65536` þá er `i * i == 0` vegna overflow og því verður deilt með 0.

b. `for (int i = 1; i <= N; i++) sum += 1.0 / i*i;`

Þetta skilar ekki réttri útkomu því að röð aðgerða í Java þýðir að fyrst verður reiknað út úr `1.0 / i` og sú útkoma margfölduð með `i`, þ.e. eins og skrifað hefði verið `sum += (1.0 / i) * i` sem þýðir að sum fær gildið N.

c. `for (int i = 1; i <= N; i++) sum += 1.0 / (i*i);`

Þetta er örlítið réttara nema að hér er deilt með 0 þegar `i == 65536` eins og í lið a. Hins vegar er þetta fleytitöludeiling og gildinu 0 er kastað í 0.0 og `1.0 / 0.0` er "talan" `Infinity` í Java samkvæmt skilgreiningu.

d. `for (int i = 1; i <= N; i++) sum += 1 / (1.0*i*i);`

Þetta skilar loksins réttu svari með 12 réttum aukastöfum.

Cloudcoder æfingar

1.2.14

```
public class DividesEvenly {
    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);

        boolean t = (a % b == 0) || (b % a == 0);
        System.out.println(t);
    }
}
```

1.3.1

Athugið að hér þarf ekki að tékka á hvort `a==c` vegna þess að ef `a==b` og `b==c` þá verður það að gilda.

```
public class AllThreeEqual {
    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = Integer.parseInt(args[2]);

        if ((a == b) && (b == c)) {
```

```
            System.out.println("equal");
        } else {
            System.out.println("not equal");
        }
    }
}
```

1.3.4

```
public class IntervalCheck {
    public static void main(String[] args) {
        double x = Double.parseDouble(args[0]);
        double y = Double.parseDouble(args[1]);

        boolean b = (0 < x) && (x < 1);
        b = b && (0 < y) && (y < 1);

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

Verkefni - 1.3.35

```
public class RandomWalk {
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    int reps = Integer.parseInt(args[1]);

    // define variables needed to keep track
    // of the experiment
    double totalDistance = 0.0; // sum of all distances

    // repeat experiment reps times
    for (int i = 0; i < reps; i++) {
      // simulate one random walk starting at (0,0)
      // until you hit the boundary
      int dist = 0;
      int x = 0, y = 0; // current position
      while (x != N && x != -N && y != N && y != -N) {
        double r = Math.random(); // random move
        if (r < 0.25) {
          x++;
        } else if (r < 0.5) {
          x--;
        } else if (r < 0.75) {
          y++;
        } else {
```

```
            y--;
          }
          dist += 1; // update distance
        }

        totalDistance += dist; // update sum of distances
      }

      // summarize the experiments
      // and print out the final results
      System.out.print("Average distance for "
                       + (2*N) + " x " + (2*N) + " cube is: ");
      System.out.println(totalDistance / reps);
    }
}
```

Ef við keyrum forritið fæst

```
% java RandomWalk 1000 1000
Average distance for 2000 x 2000 cube is: 1166249.978
```

og tekur um 33s í keyrslutíma.