10. vežbe

1. zadatak

```
/* Napisati program koji za dati ceo broj n, 0 <= n <= 40, ispisuje vrednost
   elementa rekurentnog niza f_n. Niz je definisan na sledeci nacin:
   f_n = f_n-1 - 2f_n-2, n neparno, n >= 3
   f_n = f_n-2 + 3f_n-3, n parno, n >= 3
   f_2 = 2, f_1 = 3, f_0 = 1
   Element f_n izracunati:
    (a) rekurzivno preko definicije,
    (b) rekurzivno pomocu akumulirajuceg parametra,
    (c) iterativno.
   U slucajevima (b) i (c) voditi racuna o efikasnosti resenja.
 */

class Recurrent2 {

  static final int granica = 40;
```

1. zadatak

```
// (a) rekurzivno preko definicije
static int fRec1(int n) {
  if (n == 0)
    return 1;
  else if (n == 1)
    return 3;
  else if (n == 2)
    return 2;
  else if (n % 2 == 1)
    return fRec1(n-1) - 2*fRec1(n-2);
  else
    return fRec1(n-2) + 3*fRec1(n-3);
}
```

1. zadatak

```
// (b) rekurzivno pomocu akumulirajuceg parametra
static int ff(int f2, int f1, int f0, int i, int n) {
  if (i > n)
    return f2;
  else if (i % 2 == 1)
    return ff(f2-2*f1, f2, f1, i+1, n);
  else
    return ff(f1+3*f0, f2, f1, i+1, n);
}

static int fRec2(int n) {
  if (n == 0)
    return 1;
  else if (n == 1)
    return 3;
  else if (n == 2)
    return 2;
  else
    return ff(2, 3, 1, 3, n);
}
```

1. zadatak

```
// (c) iterativno, ali prostorno neefikasno
static int fIter1(int n) {
  int f[] = new int[granica + 1];
  f[0] = 1;
  f[1] = 3;
  f[2] = 2;
  for (int i = 3; i <= n; i++) {
    if (i % 2 == 1)
      f[i] = f[i-1] - 2*f[i-2];
    else
      f[i] = f[i-2] + 3*f[i-3];
  }
  return f[n];
}
```

1. zadatak

```
// (c) iterativno, efikasno
static int fIter2(int n) {
  int fn, f0 = 1, f1 = 3, f2 = 2;
  if (n == 0)
    fn = f0;
  else if (n == 1)
    fn = f1;
  else if (n == 2)
    fn = f2;
  else {
    fn = 0;
    for (int i = 3; i <= n; i++) {
      if (i % 2 == 1)
        fn = f2 - 2*f1;
      else
        fn = f1 + 3*f0;
      f0 = f1;
      f1 = f2;
      f2 = fn;
    }
  }
  return fn;
}
```

1. zadatak

```
  public static void main(String[] args) {
    System.out.print("Unesite n (0 <= n <= " + granica + "): ");
    int n = Svetovid.in.readInt();
    if (0 <= n && n <= granica) {
      System.out.println("fRec1(n) = " + fRec1(n));
      System.out.println("fRec2(n) = " + fRec2(n));
      System.out.println("fIter1(n) = " + fIter1(n));
      System.out.println("fIter2(n) = " + fIter2(n));
    }
    else {
      System.out.println("n je van dozvoljenih granica");
    }
  }
}
```

2. zadatak

```
/* Napisati program koji za dati ceo broj n, 1 <= n <= 30, ispisuje vrednost
   elementa rekurentnog niza f_n. Niz je definisan na sledeci nacin:
   f_n = f_n-1 + g_n-1, n >= 2
   g_n = 2g_n-1 - f_n-1, n >= 2
   f_1 = 2
   g_1 = 3
   Element f_n izracunati:
    (a) rekurzivno preko definicije,
    (b) rekurzivno pomocu akumulirajuceg parametra,
    (c) iterativno.
   U slucajevima (b) i (c) voditi racuna o efikasnosti resenja.
 */

class Recurrent3 {

  static final int granica = 30;
```

2. zadatak

```
// (a) rekurzivno preko definicije
static int fRec1(int n) {
  if (n == 1)
    return 2;
  else
    return fRec1(n-1) + gRec1(n-1);
}

static int gRec1(int n) {
  if (n == 1)
    return 3;
  else
    return 2 * gRec1(n-1) - fRec1(n-1);
}
```

2. zadatak

```
// (b) rekurzivno pomocu akumulirajuceg parametra
static int fg(int f1, int g1, int i, int n) {
  if (i > n)
    return f1;
  else
    return fg(f1+g1, 2*g1-f1, i+1, n);
}

static int fRec2(int n) {
  if (n == 1)
    return 2;
  else
    return fg(2, 3, 2, n);
}
```

2. zadatak

```
// (c) iterativno, ali prostorno neefikasno
static int fIter1(int n) {
  int f[] = new int[n + 1];
  int g[] = new int[n + 1];
  f[1] = 2;
  g[1] = 3;
  for (int i = 2; i <= n; i++) {
    f[i] = f[i-1] + g[i-1];
    g[i] = 2*g[i-1] - f[i-1];
  }
  return f[n];
}
```

2. zadatak

```
// (c) iterativno, efikasno
static int fIter2(int n) {
  int fn, gn;
  int f1 = 2;
  int g1 = 3;
  if (n == 1)
    fn = f1;
  else {
    fn = 0;
    gn = 0;
    for (int i = 2; i <= n; i++) {
      fn = f1 + g1;
      gn = 2*g1 - f1;
      f1 = fn;
      g1 = gn;
    }
  }
  return fn;
}
```

2. zadatak

```
  public static void main(String[] args) {
    System.out.print("Unesite n (1 <= n <= " + granica + "): ");
    int n = Svetovid.in.readInt();
    if (1 <= n && n <= granica) {
      System.out.println("fRec1(n) = " + fRec1(n));
      System.out.println("fRec2(n) = " + fRec2(n));
      System.out.println("fIter1(n) = " + fIter1(n));
      System.out.println("fIter2(n) = " + fIter2(n));
    }
    else {
      System.out.println("n je van dozvoljenih granica");
    }
  }
}
```

3. zadatak

```
/* Napisati program koji za dati ceo broj n, 0 <= n <= 40, ispisuje vrednost
   elementa rekurentnog niza f_n. Niz je definisan na sledeci nacin:
   f_n = f_n-1 + g_n-2, n >= 2, poslednja cifra n je >= 5
   f_n = f_n-2 - g_n-1, n >= 2, poslednja cifra n je < 5
   g_n = g_n-1 - 2f_n-2, n >= 2, n neparno
   g_n = g_n-2 + 2f_n-1, n >= 2, n parno
   f_1 = 0, f_0 = -1
   g_1 = 1, g_0 = 0
   Element f_n izracunati:
    (a) rekurzivno preko definicije,
    (b) rekurzivno pomocu akumulirajuceg parametra,
    (c) iterativno.
   U slucajevima (b) i (c) voditi racuna o efikasnosti resenja.
 */

class Recurrent4 {

  static final int granica = 40;
```

3. zadatak

```
// (a) rekurzivno preko definicije
static int fRec1(int n) {
  if (n <= 1)
    return n-1;
  else if (n % 10 >= 5)
    return fRec1(n-1) + gRec1(n-2);
  else
    return fRec1(n-2) - gRec1(n-1);
}

static int gRec1(int n) {
  if (n <= 1)
    return n;
  else if (n % 2 == 1)
    return gRec1(n-1) - 2*fRec1(n-2);
  else
    return gRec1(n-2) + 2*fRec1(n-1);
}
```

3. zadatak

```
// (b) rekurzivno pomocu akumulirajuceg parametra
static int fg(int f1, int f0, int g1, int g0, int i, int n) {
  int fn, gn;
  if (i > n)
    return f1;
  else {
    if (i % 10 >= 5)
      fn = f1 + g0;
    else
      fn = f0 - g1;
    if (i % 2 == 1)
      gn = g1 - 2*f0;
    else
      gn = g0 + 2*f1;
    return fg(fn, f1, gn, g1, i+1, n);
  }
}
static int fRec2(int n) {
  if (n <= 1)
    return n-1;
  else
    return fg(0, -1, 1, 0, 2, n);
}
```

3. zadatak

```
// (c) iterativno
static int fIter(int n) {
  int f0, f1, fn, g0, g1, gn;
  fn = 0;
  if (n <= 1)
    fn = n-1;
  else {
    f0 = -1; f1 = 0; g0 = 0; g1 = 1;
    for (int i = 2; i <= n; i++) {
      if (i % 10 >= 5)
        fn = f1 + g0;
      else
        fn = f0 - g1;
      if (i % 2 == 1)
        gn = g1 - 2*f0;
      else
        gn = g0 + 2*f1;
      f0 = f1; f1 = fn; g0 = g1; g1 = gn;
    }
  }
  return fn;
}
```

3. zadatak

```
  public static void main(String[] args) {
    System.out.print("Unesite n (0 <= n <= " + granica + "): ");
    int n = Svetovid.in.readInt();
    if (0 <= n && n <= granica) {
      System.out.println("fRec1(n) = " + fRec1(n));
      System.out.println("fRec2(n) = " + fRec2(n));
      System.out.println("fIter(n) = " + fIter(n));
    } else {
      System.out.println("n je van dozvoljenih granica");
    }
  }
}
```

4. zadatak

```
/* Napisati program koji za date cele brojeve n, 0 <= n <= 50, i
   r, 2 <= r <= 20, ispisuje vrednost elementa rekurentnog niza f_n.
   Niz je definisan na sledeci nacin:
   f_n = f_n-1 - g_n-2 + f_n-r - g_n-r, n >= r
   g_n = g_n-1 + f_n-2 - f_n-r - g_n-r, n >= r
   f_n = 1, 0 <= n < r
   g_n = 2, 0 <= n < r
   Element f_n izracunati:
    (a) rekurzivno preko definicije,
    (b) rekurzivno pomocu akumulirajuceg parametra,
    (c) iterativno.
   U slucajevima (b) i (c) voditi racuna o efikasnosti resenja.
 */

class Recurrent5 {

  static final int maxN = 50;
  static final int maxR = 20;
```

4. zadatak

```
// (a) rekurzivno preko definicije
static int fRec1(int n, int r) {
  if (n < r)
    return 1;
  else
    return fRec1(n-1, r) - gRec1(n-2, r) + fRec1(n-r, r) - gRec1(n-r, r);
}

static int gRec1(int n, int r) {
  if (n < r)
    return 2;
  else
    return gRec1(n-1, r) + fRec1(n-2, r) - fRec1(n-r, r) - gRec1(n-r, r);
}
```

4. zadatak

```
// (b) rekurzivno pomocu akumulirajuceg parametra
static int fg(int f[], int g[], int i, int n, int r) {
  if (i > n)
    return f[r];
  else {
    f[r] = f[r-1] - g[r-2] + f[0] - g[0];
    g[r] = g[r-1] + f[r-2] - f[0] - g[0];
    for(int j = 0; j < r; j++) {
      f[j] = f[j+1];
      g[j] = g[j+1];
    }
    return fg(f, g, i+1, n, r);
  }
}
static int fRec2(int n, int r) {
  int f[] = new int[r + 1];
  int g[] = new int[r + 1];
  if (n < r)
    return 1;
  else {
    for (int j = 0; j < r; j++) {
      f[j] = 1; g[j] = 2;
    }
    return fg(f, g, r, n, r);
  }
}
```

4. zadatak

```
// (c) iterativno
static int fIter(int n, int r) {
  int f[] = new int[r + 1];
  int g[] = new int[r + 1];
  for (int i = 0; i < r; i++) {
    f[i] = 1; g[i] = 2;
  }
  if (n < r)
    return f[n];
  else {
    for (int i = r; i <= n; i++) {
      f[r] = f[r-1] - g[r-2] + f[0] - g[0];
      g[r] = g[r-1] + f[r-2] - f[0] - g[0];
      for (int j = 0; j < r; j++) {
        f[j] = f[j+1];
        g[j] = g[j+1];
      }
    }
  }
  return f[r];
}
```

4. zadatak

```
  public static void main(String[] args) {
    System.out.print("Unesite n (0 <= n <= " + maxN + "): ");
    int n = Svetovid.in.readInt();
    System.out.print("Unesite r (1 <= r <= " + maxR + "): ");
    int r = Svetovid.in.readInt();
    if (0 <= n && n <= maxN && 1 <= r && r <= maxR) {
      System.out.println("fRec1(n, r) = " + fRec1(n, r));
      System.out.println("fRec2(n, r) = " + fRec2(n, r));
      System.out.println("fIter(n, r) = " + fIter(n, r));
    } else {
      System.out.println("n i/ili r je van dozvoljenih granica");
    }
  }
}
```