Oplossing oefeningen Hoofdstuk 5

Oefening 1:

```
{
    // declaratie
    final int DIM = 50;   // tabel bevat 50 elementen
    int[] getal = new int[DIM];       // tabel met getallen
    int atlGetallen;      // aantal ingevoerde getallen
    int i;                // index voor doorlopen tabel getal
    char nog;     // karakter voor controle om verder te gaan

    // inbrengen alle getallen
    i = 0;
    nog = 'j';

    // zolang nog getallen en nog plaats in tabel
    while ((nog != 'n') && (i < DIM))
    {
     System.out.print("Geef een integer in: ");
     getal[i] = sc.nextInt();       // inbrengen volgend getal
     i = i + 1;
     System.out.print("Wil je nog een getal ingeven? ");
     nog = sc.next().charAt(0);
    } //end while

    // foutmelding bij dimensieoverschrijding
    if (nog != 'n')
    {
     System.out.println("Het programma is slechts geschikt voor "+DIM+
          " getallen");
    }


    // eindetabelkenmerk
    atlGetallen = i;

    // getallen omgekeerd afbeelden
    for (i = atlGetallen-1; i >= 0; i--)
    {
     System.out.print(getal[i] + " ");
    }

}
```

Oefening 2:

```
{
    // declaratie
    final int DIM = 100;              // tabel bevat 100 elementen
    int[] fibon = new int[DIM];       // tabel met fibonacci getallen
    int i;                            // index voor doorlopen fibon

    // invullen eerste 2 fibonacci getallen
    fibon[0] = 0;
    fibon[1] = 1;

    // invullen volgende elementen m.b.v. formule
    for (i = 2; i < DIM; i = i + 1)
    {
     fibon[i] = fibon[i-1] + fibon[i-2];
    }

}
```

Oefening 3:

```
{
    final int DIM = 15;               // tabel bevat 100 elementen
    int[] fibon = new int[DIM];       // tabel met fibonacci getallen
    int i;                            // index voor doorlopen fibon
    int x;                            // zoekgetal

    // inbrengen tabel met fibonacci getallen
    fibon[0] = 0;
    fibon[1] = 1;
    for (i = 2; i < DIM; i = i + 1)
    {
     fibon[i] = fibon[i-1] + fibon[i-2];
    }
    // inbrengen zoekgetal met controle
    System.out.println("Geef getal tussen "+fibon[0]+" en "+
          fibon[DIM-1]);
    x = sc.nextInt();
    while (x <= fibon[0] || x > fibon[DIM-1])
    {
     System.out.print("Ongeldig getal, geef een nieuw getal: ");
     x = sc.nextInt();
    } // end while
```

```
    // zoeken naar eerste tabelelement >= x
    i = 1;
    while (x > fibon[i])
    {
      i = i + 1;
    }

    // afbeelden resultaat
    System.out.println("Index is "+i);
}
```

Oefening 4:

```
{
    final int DIM = 51;    // tabel tab bevat 50 elementen + eindelement
    double[] tab = new double[DIM];       // tabel met getallen
    int i;                 // index voor doorlopen tabel tab
    int atlGetallen;       // aantal gebruikte tabelelementen
    double som;            // som van alle gebruikte tabelelementen
    double gemid;          // gemiddelde
    double afw;            // afwijking van tabelelement t.o.v. gemiddelde
    double standAfw;       // standaardafwijking

    // inbrengen tabel
    tab = receive();

    // tabelelementen sommeren
    som = 0;
    i = 0;
    while (tab[i] != -1)
    {
      som = som + tab[i];
      i = i + 1;
    }
    atlGetallen = i;
    // berekenen en afbeelden gemiddelde
    if (atlGetallen != 0)
      gemid = som / atlGetallen;
    else
      gemid = 0;

    System.out.println("Het gemiddelde is "+gemid);
```

```
    // sommeren van kwadratische afwijkingen t.o.v. gemiddelde
    som = 0;
    i = 0;
    while (tab[i] != -1)
    {
      afw = tab[i]-gemid;
      som = som + (afw * afw);
      i = i + 1;
    }
    // berekenen en afbeelden standaardafwijking
    if (atlGetallen != 0)
    {
      standAfw = Math.sqrt(som / atlGetallen);
    }
    else
    {
      standAfw = 0;
    }
    System.out.println("De standaardafwijking is "+standAfw);
}
```

Oefening 4: (aangepast voor BlueJ)

```
{
    double[] tab = {178,167.5,168,198,190,191,158,152.4,180,170};

    int i;              // index voor doorlopen tabel tab
    int atlGetallen;    // aantal gebruikte tabelelementen
    double som;         // som van alle gebruikte tabelelementen
    double gemid;       // gemiddelde
    double afw;         // afwijking van tabelelement t.o.v. gemiddelde
    double standAfw;    // standaardafwijking

    // tabelelementen sommeren
    som = 0;
    i = 0;
    while (i < tab.length)
    {
      som = som + tab[i];
      i = i + 1;
    }
    // berekenen en afbeelden gemiddelde
    if (tab.length != 0)
      gemid = som / tab.length;
    else
      gemid = 0;

    System.out.println("Het gemiddelde is "+gemid);
```

```
    // sommeren van kwadratische afwijkingen t.o.v. gemiddelde
    som = 0;
    i = 0;
    while (i < tab.length)
    {
     afw = tab[i]-gemid;
     som = som + (afw * afw);
     i = i + 1;
    }
    // berekenen en afbeelden standaardafwijking
    if (tab.length != 0)
    {
     standAfw = Math.sqrt(som / tab.length);
    }
    else
    {
     standAfw = 0;
    }
    System.out.println("De standaardafwijking is "+standAfw);
}
```

Oefening 5:

```
{
    final int DIM1 = 1001; // tabel tab1 met eindelement -1
    int[] tab1 = new int[DIM1];       // 1° gegeven tabel
    int i1;              // index voor doorlopen tabel tab1
    final int DIM2 = 501;  // tabel tab2 met eindelement -1
    int[] tab2 = new int[DIM2];       // 2° gegeven tabel
    int i2;              // index voor doorlopen tabel tab2
    final int DIM3 = 1501; // tabel tab3 met eindelement -1
    int[] tab3 = new int[DIM3];       // samengevoegde tabel
    int i3;              // index voor doorlopen tabel tab3

    // inbrengen gegeven tabellen
    tab1 = receive();
    tab2 = receive();

    // Positioneren tab1, tab2, tab3 op eerste element
    i1 = 0;
    i2 = 0;
    i3 = 0;
```

```
    // plaats meermaals kleinste element uit tab1 of
    // tab2 in tab3
    while (tab1[i1] != -1 && tab2[i2] != -1)
    {
      if (tab1[i1] < tab2[i2])
      {
        tab3[i3] = tab1[i1];
        i1 = i1 + 1;  // verplaatsen in tab1
      }
      else
      {
        tab3[i3] = tab2[i2];
        i2 = i2 + 1;  // verplaatsen in tab2
      }
      i3 = i3 + 1;    // verplaatsen in tab3
    }

    // plaats eventueel overschot van tab1 in tab3
    while (tab1[i1] != -1)
    {
      tab3[i3] = tab1[i1];
      i1 = i1 + 1;
      i3 = i3 + 1;
    }

    // plaats eventueel overschot van tab2 in tab3
    while (tab2[i2] != -1)
    {
      tab3[i3] = tab2[i2];
      i2 = i2 + 1;
      i3 = i3 + 1;
    }

    tab3[i3] = -1;       // plaats eindelement in tab3
}
```

Oefening 5: (aangepast voor BlueJ)

```
{
    // tabel  met 10 gesorteerde elementen
    int[] tab1 = {2,6,8,9,11,13,18,23,67,70};
    int i1;              // index voor doorlopen tabel tab1
    // tabel  met 5 gesorteerde elementen
    int[] tab2 = {3,4,7,8,25};
    int i2;              // index voor doorlopen tabel tab2
    // tabel tab3 met lengte = som van de lengtes van tab1 en tab2
    int[] tab3 = new int[tab1.length+tab2.length];
    int i3;              // index voor doorlopen tabel tab3
```

```
    // Positioneren tab1, tab2, tab3 op eerste element
    i1 = 0;
    i2 = 0;
    i3 = 0;

    // plaats meermaals kleinste element uit tab1 of
    // tab2 in tab3
    while (i1 < tab1.length && i2 < tab2.length)
    {
     if (tab1[i1] < tab2[i2])
     {
       tab3[i3] = tab1[i1];
       i1 = i1 + 1;  // verplaatsen in tab1
     }
     else
     {
       tab3[i3] = tab2[i2];
       i2 = i2 + 1;  // verplaatsen in tab2
     }
     i3 = i3 + 1;    // verplaatsen in tab3
    }

    // plaats eventueel overschot van tab1 in tab3
    while (i1 < tab1.length)
    {
     tab3[i3] = tab1[i1];
     i1 = i1 + 1;
     i3 = i3 + 1;
    }
    // plaats eventueel overschot van tab2 in tab3
    while (i2 < tab2.length)
    {
     tab3[i3] = tab2[i2];
     i2 = i2 + 1;
     i3 = i3 + 1;
    }

    //Afdrukken van tab3
    for(i3=0; i3<tab3.length; i3++)
    {
       System.out.print(tab3[i3]+" ");
    }
}
```

Oefening 6:

```
{
    final int DIM = 10;    // aantal rijen en kolommen van matrix
    int[][] mat = new int[DIM][DIM];     // vierkante matrix
    int atlRij;            // aantal gebruikte rijen
    int i;                 // rijindex
    int j;                 // kolomindex

    // inlezen aantal rijen
    System.out.println("Geef aantal rijen van vierkante matrix (max. " +
                       DIM+")");
    atlRij = sc.nextInt();
    while (atlRij < 1 || atlRij > DIM)
    {
     System.out.print("Aantal rijen tussen 1 en "+DIM);
     atlRij = sc.nextInt();
    }

    // inlezen alle matrixelementen
    // loop voor alle rijen
    for (i = 0; i < atlRij; i = i + 1)
    {
     // loop voor alle kolommen
     for (j = 0; j < atlRij; j = j + 1)
     {
      // inbrengen 1 matrixelement
      System.out.print("Geef matrixelement op rij "+(i + 1)+" en kolom "+
                       (j + 1));
      mat[i][j] = sc.nextInt();
     } // end for kolommen

    } // end for rijen
}
```

Oefening 7:

```
{
    final int DIM = 10;    // aantal rijen en kolommen van matrix
    int[][] mat = new int[DIM][DIM];     // vierkante matrix
    int atlRij;            // aantal gebruikte rijen
    int i;                 // rijindex
    int j;                 // kolomindex
    boolean symm;          // is matrix symmetrisch?

    // inbrengen aantal rijen en matrix
    atlRij = receive();
    mat = receive();
```

```
    // controle alle elementen in linkeronderhoek
    symm = true;
    // loop voor alle te controleren rijen
    for (i = 1; i < atlRij && symm; i = i + 1)
    {
      // loop voor alle te controleren kolommen
      for (j = 0; j < i && symm; j = j + 1)
      {
        // vergelijk matrixelement met symmetrie-element
        if (mat[i][j] != mat[j][i])
        {
          // aanduiden niet symmetrisch en loops verlaten
          symm = false;
          i = DIM;
          j = DIM;
        }
      } // end for kolommen
    } // end for rijen

    // output
    if (symm == true)
    {
      System.out.println("Matrix symmetrisch");
    }
    else
    {
      System.out.println("Matrix niet symmetrisch");
    }
}
```

Oefening 7 (aangepast voor BlueJ):

```
{
       int[][] mat = {{4,6,7,8,9},
                      {6,1,4,5,9},
                      {7,4,5,3,2},
                      {8,5,3,1,0},
                      {9,9,2,0,6}}; // vierkante matrix

    int i;              // rijindex
    int j;              // kolomindex
    boolean symm;       // is matrix symmetrisch?
```

```
    // controle alle elementen in linkeronderhoek
    symm = true;
    // loop voor alle te controleren rijen
    for (i = 1; i < mat.length && symm == true; i = i + 1)
    {
      // loop voor alle te controleren kolommen
      for (j = 0; j < i && symm == true; j = j + 1)
      {
        // vergelijk matrixelement met symmetrie-element
        if (mat[i][j] != mat[j][i])
        {
          // aanduiden niet symmetrisch en loops verlaten
          symm = false;
          i = mat.length;
          j = mat.length;
        }
      } // end for kolommen
    } // end for rijen

    // output
    if (symm == true)
    {
      System.out.println("Matrix symmetrisch");
    }
    else
    {
      System.out.println("Matrix niet symmetrisch");
    }
}
```

Oefening 8:

```
{
    final int ATLWERKN = 8;             // aantal werknemers
    final int ATLMAANDEN = 12;          // aantal maanden
    double[][] loon = new double[ATLWERKN][ATLMAANDEN]; //maandlonen
    double[] jaarloon = new double[ATLWERKN]; // jaarlonen van alle werknr.
    int[] werknr = new int[ATLWERKN]; // indextabel met volgnrs werknr.
    double somLoon;     // som van maandlonen
    int i;              // rijindex
    int j;              // kolomindex
    int g;              // index van grootste element van
                        // ongesorteerd deel
    int hulp;           // hulpvariabele om te verwisselen
    double dHulp;       // double hulpvar om te verwisselen
```

```
    // inbrengen 2-dimensionale tabel loon
    loon = receive();

    // loop van alle werknemers
    for (i = 0; i < ATLWERKN; i = i + 1)
    {
      // sommeren alle maandlonen van 1 werknemer
      somLoon = 0;
      for (j = 0; j < ATLMAANDEN; j = j + 1)
        somLoon = somLoon + loon[i][j];

      // opbouwen sorteertabel en indextabel
      jaarloon[i] = somLoon;
      werknr[i] = i;

    } //end for werknemers

    // sorteren jaarlonen in dalende volgorde via indexering
    for (i = 0; i < ATLWERKN-1; i = i + 1)
    {
      // zoek het grootste element uit ongesorteerd deel
      g = i;
      for (j = i + 1; j < ATLWERKN; j = j + 1)
      {
        if (jaarloon[j] > jaarloon[g])
          g = j;
      }

      // verwissel in indextabel
      hulp = werknr[i];
      werknr[i] = werknr[g];
      werknr[g] = hulp;
      //verwissel in jaarloontabel
      dHulp = jaarloon [i];
      jaarloon [i] = jaarloon [g];
      jaarloon [g] = dHulp;

    } // end for

    // output
    for (i = 0; i < ATLWERKN; i = i + 1)
    {
      System.out.println("Werknemer "+werknr[i] + 1+" heeft jaarloon "+
            jaarloon[i]);
    }
}
```

Oefening 9:

```
{
    int[][] mat = {{1,8,3},
                   {6,4,2},
                   {5,0,7}};     // vierkante matrix

    int i;               // rijindex
    int j;               // kolomindex
    int som;             // som van 1 rij, kol of diagonaal
    int refSom;          // referentiesom om te vergelijken
    boolean magisch;     // is matrix symmetrisch?


    // initialiseer magisch en bereken referentiesom
    // als eerste rijsom
    magisch = true;
    refSom=0;
    for (j = 0; j < mat.length; j++)
    {
     refSom = refSom + mat[0][j];
    }



    // vergelijk volgende rijsommen met referentiesom
    for (i = 1; i < mat.length; i++)
    {

      // bereken rijsom
      som = 0;
     for (j = 0; j < mat.length; j++)
     {
       som = som + mat[i][j];
     }



     // vergelijk rijsom met referentiesom
     if (som != refSom)
     {
       // aanduiden niet magisch en rijcontrole verlaten
       magisch = false;
       i = mat.length;
     }

    } // end for
```

```
// vergelijk kolomsommen met referentiesom
if (magisch == true)
{
  for (j = 0; j < mat.length; j++)
  {
    // bereken kolomsom
     som = 0;
    for (i = 0; i < mat.length; i++)
    {
      som = som + mat[i][j];
    }

    // vergelijk kolomsom met referentiesom
    if (som != refSom)
    {
      // aanduiden niet magisch en kolomcontrole verlaten
      magisch = false;
      j = mat.length;
    }
  } // end for
} // end if


// vergelijk hoofddiagonaal met referentiesom
if (magisch == true)
{
  // bereken som hoofddiagonaal
   som = 0;
  for (i = 0; i < mat.length; i++)
  {
    som = som + mat[i][i];
  }
  // vergelijk som hoofddiagonaal met referentiesom
  if (som != refSom)
  {
    magisch = false;
  }
} // end if


// vergelijk nevendiagonaal met referentiesom
if (magisch == true)
{
  // bereken som nevendiagonaal
   som = 0;
  for (i = 0; i < mat.length; i++)
  {
    j = mat.length-i-1;
    som = som + mat[i][j];
  }
```

```
        // vergelijk som nevendiagonaal met referentiesom
        if (som != refSom)
        {
          magisch = false;
        }
      } // end if

      // output
      if (magisch == true)
      {
        System.out.println("Matrix is magisch vierkant");
      }
      else
      {
        System.out.println("Matrix is geen magisch vierkant");
      }
}
```