Tentamen Imperatief Programmeren

Maandag 5 november 2007, 9:00–12:00 uur, examenhal

- Schrijf boven ieder blad je naam, studentnummer, en volgnummer van het blad. Schrijf op het eerste blad het aantal ingeleverde bladen.
- Lees eerst een opgave volledig door alvorens deze te maken.
- Schrijf netjes en zorgvuldig met een pen (geen potlood).
- Je hebt 3 uur de tijd. Gebruik deze nuttig. Als je snel klaar bent, gebruik dan de resterende tijd om je antwoorden nog eens te controleren.
- Succes!

Opgave 1: Toekenningen
Bepaal voor ieder van de onderstaande annotaties de keuze die op de plaats van de lege regel (.....) ingevuld kan worden. Per onderdeel is er precies één keuze mogelijk. De variabelen `x`, `y` en `h` zijn van het type `int`. Let erop dat `X` en `Y` (met hoofdletter!) specificatie-constanten (en dus geen variabelen) zijn.

1.1
```
// x==X+1
.....
// x==3*X + 4
```
```
(a) x=3*(x-1)+1;
(b) x=3*(X-1)+4;
(c) x=3*x+1;
```

1.2
```
// x==2*X+Y, y==Y
.....
// x==X
```
```
(a) x=X;
(b) x=(x-y)/2;
(c) x=x/2 - y;
```

1.3
```
// x-3==3*X
.....
// x==X
```
```
(a) x=x/3-1;
(b) x=(x+3)*3;
(c) x=3*(x-3);
```

1.4
```
// x==X, y==Y
y=x; x=y;
.....
```
```
(a) // x==Y, y==X
(b) // x==X, y==X
(c) // x==Y, y==Y
```

1.5
```
// x==X, y==Y
x=x-y; y=x+y; x=y-x;
// .....
```
```
(a) // x==Y, y==X
(b) // x==X, y==Y
(c) // x==X, y==X
```

1.6
```
// x==X, y==Y+1
x=x-y; y=x+y; x=y-x;
.....
```
```
(a) // x==X+1, y==X
(b) // x==X, y==Y+1
(c) //  x==Y+1, y==X
```

Opgave 2: Vermenigvuldigen van grote getallen
Als je in JAVA grote gehele getallen met elkaar vermenigvuldigt dan zul je ontdekken dat het resultaat niet altijd klopt. Een `int` word gerepresenteerd door 32 bits, terwijl het resultaat van de vermenigvuldiging soms meer bits nodig heeft.

Het onderstaande programma (zie volgende vel) lost dit probleem op door getallen te representeren door arrays van cijfers. Het vermenigvuldigen gaat ongeveer volgens het algoritme dat je op de basisschool hebt geleerd. Het programma bevat echter 5 fouten. Geef voor iedere fout een correctie.

```
1   class Vermenigvuldiger {
2       int[] cijfersA, cijfersB, cijfersC;
3
4       void converteer(int[] rij, String s) {
5         for (int i=0; i<s.length(); i++) {
6           rij[i] = s.charAt(s.length()-1-i) - "0";
7         }
8         for (int i=s.length(); i<rij.length; i++) {
9           rij[i] = 0;
10        }
11      }
12
13      void initialiseer(String a, String b) {
14          int[] cijfersA, cijfersB, cijfersC;
15          cijfersA = new int[a.length()];
16          cijfersB = new int[b.length()];
17          cijfersC = new int[a.length()+b.length()];
18          converteer(cijfersA, a);
19          converteer(cijfersB, b);
20          converteer(cijfersC, "0");
21      }
22
23      void toonCijfers(int[] rij) {
24          int i=rij.length-1;
25          while ((i>0) && (rij[i]==0)) { // negeer vooroplopende nullen
26             i--;
27          }
28          while (i>=0) {
29             System.out.print(rij[i]);
30          }
31      }
32
33      void vermenigvuldig() {
34         for (int i=0; i<cijfersB.length; i++) {
35           for (int j=0; j<cijfersB.length; j++) {
36             int k = i+j;
37             int onthouden = cijfersA[i]*cijfersB[j];
38             while (onthouden>0) {
39               cijfersC[k] += onthouden;
40               onthouden = cijfersC[k]/10;
41               cijfersC[k] = cijfersC[k]%10;
42               k++;
43             }
44           }
45         }
46      }
47
48      public Vermenigvuldiger() {
49         initialiseer("1234567890", "987654321");
50         vermenigvuldig();
51         toonCijfers();
52         System.out.println();
53      }
54
55      public static void main(String[] args) {
56         new Vermenigvuldiger();
57      }
58  }
```

Opgave 3: Tijdscomplexiteit: time-management is ingewikkeld ...
Geef van ieder van de volgende programmafragmenten aan wat de scherpste bovengrens is (in termen van `N`) voor het aantal rekenstappen dat het fragment uitvoert. M.a.w. een algoritme dat N stappen doet is $O(N)$ en niet $O(N^2)$ omdat $O(N)$ de scherpste bovengrens is.

3.1
```
int som=0;
for (int i=0; i<=N; i++) {
 som += i;
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.2
```
int i=0;
while (i*i<=N) {
 i++;
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.3
```
int som=0;
for (int i=2*N; i>=0; i--) {
 for (int j=1; j<N; j*=2) {
  som++;
 }
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.4
```
int som=0;
for (int i=1; i<N; i*=3) {
 som += i;
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.5
```
int som=0;
for (int i=0; i<N; i+=2) {
 for (int j=i+1; j<N; j+=2) {
  som += j;
 }
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.6
```
int j=0, som=0;
for (int i=0; i<=N; i++) {
 j += i;
}
for (int i=0; i<=j; i++) {
 som += i;
}
```

(a) $O(logN)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

Opgave 4: SMS hulpje

Het versturen van boodschappen met de mobiele telefoon (SMS) heeft in de laatste 10 jaren een grote vlucht genomen. Vooral jeugdige bezitters van mobiele telefoons hebben er een ware sport van gemaakt om elkaar met SMSjes te bestoken en langzaam maar zeker ontstaat de cultuur dat men aan het versturen van een SMS rechten denkt te kunnen ontlenen. Zo stuurde een student de mededeling dat hij ziek was naar de docent van het vak Imperatief Programmeren. Helaas bleek de docent nogal ouderwets en niet in staat (bereid) om SMSjes te beantwoorden. Het schrijven van volledige zinnen tekst m.b.v. een toetsenbordje dat slechts 10 cijfers bevat is een hele toer en vereist een lenigheid van de vingers die menig kroegpianist niet bezit.

In deze opgave wordt je gevraagd de docent te helpen door een programmaatje te maken dat teksten vertaalt naar een reeks toetsaanslagen. Een tekst zoals "`ik ben ziek`" wordt omgezet in `4(3)5(2)1(1)2(2)3(2)6(2)1(1)9(4)4(3)3(2)5(2)`. Dit betekent dat de toets `4` drie keer ingedrukt dient te worden om de letter `i` te krijgen. Vervolgens 2 keer de toets `5` enz. Merk ook op dat op deze telefoon een spatie geproduceerd wordt met de '1'-toets.

Een deel van het programma `SMShulpje` is al voor je geïmplementeerd. Zie hiervoor de volgende pagina.

Het programma leest regels tekst van de invoer en zet deze om tot een reeks van losse lettertekens in het array `letters`. De integer variabele `lengte` geeft de lengte van de tekst weer in dit array, m.a.w. `letters[i]` is een goed gedefinieerd letterteken voor $0 \leq$ `i` $<$ `lengte`. Merk overigens op dat dit programma stopt als op de invoer een regel word aangetroffen die begint met het teken '#'.

(a) De methode `maakHoofdletters` vervangt alle kleine letters in het array `letters` door de corresponderende hoofdletters. Schrijf deze methode.

(b) De methode `schoonOp` verwijdert alle tekens uit het array `letters` die niet een spatie of een hoofdletter zijn. Schrijf deze methode.

(c) De methode `filterSpaties` vervangt vervolgens alle meervoudige voorkomens van spaties door enkele spaties. Als de tekst eindigt met één of meer spaties, dan dienen deze spaties volledig verwijderd te worden, m.a.w. de tekst "`AAP  NOOT MIES  `" moet omgezet worden tot "`AAP NOOT MIES`". Schrijf deze methode.

(d) De methode `vertaal` verzorgt uiteindelijk de vertaling van de gefilterde tekst tot een reeks van toetsen van de mobiele telefoon. Deze methode drukt dus bij een tekst zoals "`IK BEN ZIEK`" de reeks `4(3)5(2)1(1)2(2)3(2)6(2)1(1)9(4)4(3)3(2)5(2)` af. Schrijf deze methode. Maak hierbij eventueel gebruik van het array `toets`, zoals die is gegeven in de programmatekst. De waarde `toets[i]` geeft (voor $0 \leq$ `i` < 10) het letterteken weer dat op je mobiele telefoon verschijnt als je de toets met cijfer i indrukt. Merk op dat in Java geldt dat `' ' < 'A'`.

Het programma SMShulpje:

```
 1  import java.util.Scanner;
 2
 3  class SMShulpje {
 4      Scanner sc = new Scanner(System.in);
 5      char [] letters;
 6      int lengte;
 7
 8      String leesRegel() {  // lees een gehele regel tekst van de invoer
 9        return sc.nextLine();
10      }
11
12      void converteer(String tekst) { // zet tekst om in losse letters
13          lengte = tekst.length();
14          letters = new char[lengte];
15          for (int i=0; i<lengte; i++) {
16              letters[i] = tekst.charAt(i);
17          }
18      }
19
20      void maakHoofdletters() {
21        .......
22      }
23
24      void schoonOp() {
25        .......
26      }
27
28      void filterSpaties() {
29        .......
30      }
31
32      void vertaal() {
33        char [] toets = {' ', ' ', 'A', 'D', 'G', 'J', 'M', 'P', 'T', 'W'};
34        ......
35      }
36
37      public SMShulpje() {
38          String tekst;
39
40          tekst = leesRegel();
41          while (tekst.charAt(0) != '#') {
42              converteer(tekst);
43              maakHoofdletters();
44              schoonOp();
45              filterSpaties() {
46              vertaal();
47              tekst = leesRegel();
48          }
49      }
50
51      public static void main(String[] args) {
52         new SMShulpje();
53    }
54  }
```

Opgave 5: Iteratief/recursief zoeken
De onderstaande methode `lineairZoeken(int[] rij, int waarde)` zoekt het getal `waarde` in de rij `rij`. Als `waarde` op positie `i` in de rij voorkomt, dan retourneert de methode de index `i`. Als `waarde` niet in de rij voor komt dan retourneert de methode de waarde -1.

```
int recursiefLineairZoeken(int[] rij, int index, int waarde) {
  if (index == rij.length) {
    return -1;
  }
  if (rij[index] == waarde) {
    return index;
  }
  return recursiefLineairZoeken(rij, index+1, waarde);
}

int lineairZoeken(int[] rij, int waarde) {
  return recursiefLineairZoeken(rij, 0, waarde);
}
```

(a) De methode maakt gebruik van recursie (via `recursiefLineairZoeken`. Schrijf een iteratieve (niet-recursieve) versie van de methode `lineairZoeken(int[] rij, int waarde)`.

De onderstaande methode `binairZoeken(int[] rij, int waarde)` maakt gebruik van extra kennis, namelijk dat `rij` gesorteerd is in oplopende volgorde.

```
int binairZoeken(int[] rij, int waarde) {
  int laag = 0;
  int hoog = rij.length;
  while (laag < hoog) {
    int midden = (laag + hoog)/2;
    if (rij[midden] < waarde) {
      laag = midden + 1;
    } else {
      hoog = midden;
    }
  }
  if ((laag < rij.length) && (rij[laag] == waarde)) {
    return laag;
  } else {
    return -1;
  }
}
```

(b) Schrijf een recursieve (niet-iteratieve) versie van de methode `binairZoeken(int[] rij, int waarde)`.