Tentamen Imperatief Programmeren

Dinsdag 31 oktober 2006, 14:00-17:00, examenhal

- 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 en y zijn van het type `int`. Let erop dat `X` en `Y` (met hoofdletter!) specificatie-constanten (en dus geen variabelen) zijn.

1.1
```
// x==X+7
.....
// x==2*X + 4
```
```
(a) x=2*x-3;
(b) x=2*X-10;
(c) x=2*x-10;
```

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

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

1.4
```
// x==X+3*Y, y==2*X+2
.....
// x==6*Y, y==2*X+2
```
```
(a) x=2*x-y+2;
(b) x=2*x-y-2;
(c) x=2*(x-y/2)+1;
```

1.5
```
// x==2*X+3*Y, y==X
.....
// x==X, y==Y
```
```
(a) y=(x-2*y)/3; x=(x-3*y)/2;
(b) y=(x-3*y)/3; x=(x-2*y)/2;
(c) x=(x-2*y)/3; y=(x-3*y)/2;
```

1.6
```
// x==3*X+Y-1, y==Y
x=x+1-3*y;
.....
```
```
(a) // x==3*X-2*Y, y==Y
(b) // x==3*X-2*Y, y==X
(c) // x+1==3*X-2*Y, y==X
```

1.7
```
// x==X, y==Y
x=y; y=x;
.....
```
```
(a) // x==Y, y==X
(b) // x==X, y==X
(c) // x==Y, y==Y
```

1.8
```
// x==2*(X+Y), y==Y
x=x+y; y=x-y; x=x-y+3;
.....
```
```
(a) // x==Y-3, y==2*X+2*Y
(b) // x==Y, y-3==2*(X+Y)
(c) // x==Y+3, y==2*X+2*Y
```

1.9
```
// x==Y, y==X
x=x-y; y=x+y; x=x-y; y--; x++;
.....
```
```
(a) // x==Y-1, y==1-X
(b) // x==1+X, y==Y-1
(c) // x==1-X, y==Y-1
```

1.10
```
// x==X+Y, y==X
x=x+y; y=x-y; x=x-y;
.....
```
```
(a) // x==X, y==X+Y
(b) // x==Y, y==X-Y
(c) // x==X-Y, y==X
```

Opgave 2: als Pasen en Pinksteren op één dag vallen...

Het christelijke paasfeest begint op de eerste zondag na de eerste volle maan vanaf het begin van de lente (20/21 maart). Pasen kan in een periode van 34 dagen vallen, van 22 maart tot 25 april.

De bepaling van de exacte data is nogal gecompliceerd. Het onderstaande programma Pasen kan hierbij uitkomst bieden. Echter, het programmafragment bevat 5 fouten en is daardoor onbruikbaar. Geef aan wat fout is, en geef een correctie. Je mag ter verduidelijking gebruik maken van de regelnummers.

Let op: Je hoeft je niet bezig te houden met de ingewikkelde berekening in de programmaregels 29 tot en met 37 om de juiste maand en dag te bepalen. Je mag er op vertrouwen dat deze regels juist zijn.

```
1  import java.util.Scanner;
2
3  class Pasen {
4     Scanner sc = new Scanner(System.in);
5
6     int dag, maand;
7
8     char strMaand (int maand) {
9        if (maand = 3) {
10          return "maart";
11       }
12       return "april";
13    }
14
15    void drukAf () {
16       System.out.println ("Paaszondag: " + dag + " " + strMaand(maand));
17       int d = dag+1;
18       if (d == 32) { // Maart heeft 31 dagen
19          d = 1;
20       }
21       System.out.println ("Paasmaandag: " + d + " " + strMaand(maand));
22    }
23
24    public Pasen () {
25       int dag, maand, jaar;
26       int c, n, i, j, k, l;
27       System.out.print("Jaartal?: ");
28       jaar = sc.nextInt();
29       c = jaar/100;
30       n = jaar%19;
31       k = (c - 17)/25;
32       i = (c - c/4 - (c - k)/3 + 19*n + 15)%30;
33       i = i - (i/28)*(1 - (i/28)*(29/(i + 1)) *((21 - n)/11));
34       j = (jaar + jaar/4 + i + 2 - c + c/4)%7;
35       l = i - j;
36       maand = 3 + (l + 40)/44;
37       dag = l + 28 - 31*(maand/4);
38    }
39
40    public static void main(String[] args) {
41       new Pasen();
42    }
43 }
```

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*N; i++) {
 som += i;
}
```

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

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

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

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

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

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

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

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

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

3.6
```
int som=0;
for (int i=0; i<N; i++) {
 for (int j=i; j>0; j=j/2) {
  som++;
 }
}
```

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

Opgave 4: Radiostilte

Teun is een student met een krappe studiebeurs. Het aanschaffen van cd's is een nogal prijzige bezigheid. Daarom besluit hij zijn eigen cd's te maken met behulp van zijn pc waarmee hij digitale radio kan ontvangen via het internet. Hij luistert graag naar SkyRadio omdat dit kanaal geen irritante kletsende dj's heeft. Helaas is er wel (veel) reclame. Nu zijn de mooiste uitzendingen op maandagochtend. Helaas heeft Teun dan het vak 'Imperatief Programmeren'. Gelukkig heeft hij de eerste colleges van het vak al gevolgd en is hij nu in staat om het opnemen van muziek te automatiseren m.b.v. een Java-programmaatje, zodat hij gewoon muziek op kan nemen terwijl hij naar college gaat.

Digitale radio bestaat uit niets meer dan een stroom van integers, de zgn. samples. Iedere sample is een integer, die naar de geluidskaart wordt gestuurd en verantwoordelijk is voor een toon gedurende 1/44100e seconde. Er zijn dus 44100 samples per seconde. Teun neemt voor drie uur muziek op. Hij doet dit m.b.v. het programma `Recorder`. Een deel van het programma `Recorder` is al voor je geïmplementeerd:

```
1  import java.util.Scanner;
2
3  class Recorder {
4      Scanner sc = new Scanner(System.in);
5
6      int [] samples = new int[44100*3*3600]; // 3 uur samples
7      int volgende, lengte;
8
9      void initialiseer () { // maak een lege opname
10         volgende = 0;
11         lengte = 0;
12     }
13
14     void voegToe(int s) { // voeg sample s toe aan de opname
15         samples[volgende] = s;
16         volgende++;
17         lengte++;
18     }
19
20     public static void main(String[] args) {
21         new Recorder();
22     }
23 }
```

De samples worden opgeslagen in het array `samples`. Een toe te voegen sample wordt opgeslagen op positie `volgende`, waarna `volgende` wordt opgehoogd. Het totaal aantal samples wordt opgeslagen in de variabele `lengte`.

(a) Een geluidssample is een niet-negatief geheel getal (m.a.w. een `int` met waarde ≥ 0). Stilte bestaat uit samples met de waarde 0. We veronderstellen dat muzieknummers en commercials gescheiden worden door minstens één seconde stilte, m.a.w. een serie van minstens 44100 samples met de waarde 0. Schrijf nu een methode `int stilteFilter()` die de opname als volgt verandert. Van iedere serie van minstens 44100 nullen op rij, vervang de eerste nul door -1, om aan te geven dat hier een nieuw nummer begint. Verwijder de overige nullen uit zo'n serie door de rest van de samples op te schuiven. De methode levert uiteindelijk het totaal aantal nummers (tracks) op.

(b) Nadat de methode `stilteFilter` is toegepast houden we tracks (nummers) over. Van iedere track willen we de beginpositie kunnen vinden. Schrijf een methode `int findTrack(int nr)` die de beginpositie (index) van track `nr` oplevert. De eerste track heeft nummer 1.

(c) Een aantal van de tracks die overblijven na gebruik van `stilteFilter` zijn commercials. We nemen aan dat een track van maximaal 60 seconden een commercial is. Maak nu een methode `void printTracks(void)` die van alle niet-commercial tracks de begin- en eindpositie afdrukt.

Opgave 5: Dobbelen

Piet en Jan gaan ieder weekend gezamenlijk naar de kroeg. Alvorens ze naar de kroeg vertrekken, spelen ze een eenvoudig spel met één dobbelsteen.

Piet mag vijf keer gooien. Het totaal aantal ogen van de worpen wordt opgeteld. Vervolgens is de beurt aan Jan die eveneens vijf keer gooit. Als Piet nu 15 punten heeft gescoord en Jan heeft geen 15 punten, dan betaalt Jan de kosten van deze avond. Andersom, als Jan 15 punten heeft gescoord en Piet heeft geen 15 punten, dan betaalt Piet de kosten van deze avond. Als beiden 15 punten scoren, dan worden de kosten van de avond eerlijk gedeeld. Als geen van beiden 15 punten scoren, dan herhalen ze het spel.

(a) Schrijf een recursieve methode `aantal` die bij de aanroep `aantal(n,p)` het aantal mogelijke worpcombinaties oplevert die in `n` worpen het totaal van `p` punten oplevert. Bijvoorbeeld, `aantal(1,6)` dient 1 op te leveren, immers je kunt maar op één manier in één worp 6 punten halen. Echter, `aantal(2,3)` dient 2 op te leveren omdat de worpvolgordes $\{1,2\}$ en $\{2,1\}$ allebei 3 opleveren.

(b) Schrijf een methode `kans` die bij de aanroep `kans(n,p)` de kans afdrukt dat je daadwerkelijk `p` punten gooit in `n` worpen. Deze kans dient in gehele procenten (afgerond) afgedrukt te worden. Bijvoorbeeld, `kans(1,6)` dient 17 af te drukken.

Opgave 6: Caesar's brieven

Julius Caesar en Cleopatra correspondeerden veelvuldig. Ze hadden een bode die hun brieven bezorgde. Met name Caesar was erg wantrouwig en vermoedde dat de bode de brieven stiekem las. Om te voorkomen dat de bode de tekst begreep, besloot Caesar de brieven in het vervolg gecodeerd te versturen. Hij codeerde volgens de volgende eenvoudige methode.

Laat L de lengte van de ongecodeerde tekst (`String`) zijn. Caesar bepaalde vervolgens het kleinste gehele getal l zodanig dat $L \leq l * l$. Vervolgens schreef hij de ongecodeerde tekst op een vel papier van $l \times l$ vakjes (een matrix van $l \times l$ characters). Als de matrix niet geheel gevuld werd, dan werd deze aangevuld met spaties. Daarna bepaalde hij uit deze matrix een nieuwe $l \times l$ matrix, de zgn. transpositie, die verkregen wordt door het teken op positie i, j te verwisselen met het teken op positie j, i. Vervolgens schreef hij de tekst in gecodeerde vorm op door de transpositiematrix regel voor regel over te schrijven. Natuurlijk legde Caesar aan Cleopatra zijn systeem uit, zodat zij de boodschap weer kon decoderen.

Voorbeeld: de tekst "AVE␣CLEOPATRA" wordt via deze procedure omgezet tot de tekst "ACPAVLA␣EET␣␣OR␣". Hierin stelt het teken ␣ een spatie voor. De onderstaande matrixomzetting kan je helpen om dit te begrijpen.

A	V	E	␣
C	L	E	O
P	A	T	R
A	␣	␣	␣

A	C	P	A
V	L	A	␣
E	E	T	␣
␣	O	R	␣

Historici hebben een aantal van deze brieven gevonden, en hadden de codering snel gekraakt. Echter, het is wel monnikenwerk om met de hand deze boodschappen te decoderen. Je wordt daarom gevraagd om een methode `vertaler` te schrijven die de gebruiker vraagt of hij een tekst wil coderen of decoderen. Vervolgens wordt een regel tekst gevraagd die de methode moet coderen of decoderen. Uiteraard dient de methode de resulterende tekst af te drukken.