Tentamen Imperatief Programmeren

vrijdag 11 februari 2011, 9:00-12:00

- Schrijf boven ieder blad je naam, studentnummer, studierichting en volgnummer van het blad. Schrijf op het eerste blad het totaal aantal ingeleverde bladen.
- Je kunt in totaal 100 punten verdienen. De eerste 10 punten krijg je cadeau. Bij iedere opgave staat zijn puntenwaardering vermeld.
- 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.
- Het is niet toegestaan om een mobiele telefoon te gebruiken. Ook niet als rekenmachine!
- Succes!

Opgave 1: Toekenningen (20 punten)
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 `z` zijn van het type `int`. Let erop dat `X` en `Y` (met hoofdletter!) specificatie-constanten (en dus geen variabelen) zijn.

1.1
```
/* x + 7 == X */
.....
/* x == 5*X + 2 */
```

```
(a) x = 5*(x-7) + 2;
(b) x = 5*x + 37;
(c) x = 5*x + 35;
```

1.2
```
/* 3*x + 5*y == X */
.....
/* x + 2*y == X */
```

```
(a) x = x/3 - 3*y;
(b) x = 3*x - 3*y;
(c) x = 3*x + 3*y;
```

1.3
```
/* x + Y == 2*X, y + z == Y */
x = x + y; y = y - z;
.....
```

```
(a) /* x + z == 2*X, y + 2*z == Y */
(b) /* x == 2*X, y == Y */
(c) /* x + z == X, y + 2*z == Y */
```

1.4
```
/* x == X, y == X - Y */
x = x - y; y = y - x;
.....
```

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

1.5
```
/* x - y == Y, y == X */
y = x - y; x = x - y;
.....
```

```
(a) /* x == Y, y == X */
(b) /* x == X, y == Y */
(c) /* x == Y, y == Y */
```

1.6
```
/* x == Y, y == X */
x = x + y; y = x - y; x = x - y;
.....
```

```
(a) /* x == X, y == Y */
(b) /* x == X, y == X */
(c) /* x == Y, y == X */
```

Opgave 2: Zoek de 5 fouten (10 punten)

Het onderstaande programma sorteert een rij getallen en drukt vervolgens de gesorteerde rij af. Het programma bevat 5 fouten. Zoek de fouten en geef voor iedere fout een correctie. Let op: Fouten die met elkaar samenhangen worden beschouwd als één fout!

```
 1 #include <stdio.h>
 2
 3 void verwissel(int a, int b) {
 4   int h = a;
 5   a = b;
 6   b = h;
 7 }
 8
 9 void sorteer(int lengte, int rij[]) {
10   int i, j;
11   for (i=lengte - 1; i > 0; i--) {
12     for(j=0; j <= i; j++) {
13       if (rij[j] > rij[j+1]) {
14         verwissel(rij[j], rij[j+1]);
15       }
16     }
17   }
18 }
19
20 void toonRij(int rij[]) {
21   printf("%d", rij[0]);
22   for (i=1; i<lengte; i++) {
23     printf(" %d", rij[i]);
24   }
25   printf("\n");
26 }
27
28 int main() {
29   int a[12] = {3, 6, 1, 8, 3, 2, 9, 4, 8, 6, 5, 2};
30
31   sorteer(12, a);
32
33   printf("Na sortering: ");
34   toonRij();
35
36   return 0;
37 }
```

Opgave 3: Tijdscomplexiteit (20 punten)
In deze opgave is N een parameter (positief en geheeltallig). Geef van ieder van de volgende programmafragmenten aan wat de scherpste bovengrens is voor het aantal rekenstappen dat het fragment uitvoert in termen van N. Een algoritme dat N stappen doet is dus $O(N)$ en niet $O(N^2)$ omdat $O(N)$ de scherpste bovengrens is.

1.
```
int i, som = 1;
for (i = 1; i < N; i++) {
  som += i;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

2.
```
int i, som = 1;
for (i = 1; som < N; i++) {
  som += i;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

3.
```
int i, j, som = 0;
for (i = 1; i < N; i++) {
  for (j = i+1; j < N; j++) {
    som += j;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

4.
```
int i, j, som = 0;
for (i = 1; i < N; i+=10) {
  for (j = 0; j < 10; j++) {
    som += j;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

5.
```
int i=1;
while (i < N) {
  i = i*2;
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

6.
```
int i, j, som=0;
for (i=0; i < N; i++) {
  j = i*i;
  while (j > 0) {
    som++;
    j = j / 2;
  }
}
```
(a) $O(\log N)$ (b) $O(\sqrt{N})$ (c) $O(N)$ (d) $O(N \log N)$ (e) $O(N^2)$

Opgave 4: Valentijnsdag (20 punten)
Zoals je weet is het ieder jaar op 14 februari Valentijnsdag.

Over het ontstaan van Valentijnsdag bestaan verschillende theorieën. Waarschijnlijk ligt de oorsprong bij Sint Valentijn. Een jonge Romein die werd gemarteld maar weigerde om zijn christelijk geloof op te geven. Hij stierf op 14 februari 269 na Christus. De legende zegt dat Valentijn een dag voor zijn executie een afscheidsbriefje achterliet voor de dochter van een medegevangene waarop hij verliefd was geworden. Hij ondertekende het briefje met 'Van jouw Valentijn'.

Dit jaar (2011) valt Valentijnsdag op een maandag. We vragen ons af welke dag het was op 14 februari 269. Schrijf een functie `void toonValentijnsdag(int jaartal)` die, gegeven `jaartal`, uitrekent welke dag het was op 14 februari van het betreffende jaar. We gaan er hierbij vanuit dat `0<=jaartal` en tevens `jaartal <= 2011`. Tevens gaan we er vanuit dat de thans geldende (Gregoriaanse) kalender toen ook geldig was (in werkelijkheid gebruikte men voor 1700 de z.g.n. Juliaanse kalender, maar dat negeren we dus in deze opgave).

Hier volgen enige gegevens die je mag gebruiken:

- 14 februari 2011 is een maandag.
- De maanden april, juni, september en november hebben 30 dagen.
- De maanden januari, maart, mei, juli, augustus, oktober en december hebben 31 dagen.
- De maand februari heeft 28 dagen, behalve in schrikkeljaren.
- In een schrikkeljaar heeft de maand februari 29 dagen.
- Een jaar is een schrikkeljaar als het jaartal deelbaar is door 4, maar niet door 100. Uitzonderingen zijn jaartallen die deelbaar zijn door 400: dit zijn wel schrikkeljaren.
- Een gewoon jaar heeft 365 dagen. Een schrikkeljaar jaar heeft 366 dagen.

Opgave 5: bitrijen (20 punten)
Bekijk alle verschillende rijtjes met lengte 16 die bestaan uit enen en nullen. In totaal zijn dat er $2^{16} = 65536$. Dat is niet moeilijk; op elke positie heb je een keuze uit twee mogelijkheden. Maar hoeveel zijn het er als je als extra voorwaarde stelt dat er geen twee nullen achter elkaar mogen staan? Dat is veel lastiger. Dat blijken er 2584 te zijn.

Schrijf een routine `int aantalRijtjes(int lengte)` die voor de gegeven `lengte` het aantal verschillende rijtjes van die lengte bepaalt waarin er geen twee nullen achter elkaar mogen staan.

[Hint: het is handig een recursieve hulpfunctie `int recAantal(int len, int start)` te maken die het aantal verschillende 'goede' rijtjes met lengte `len` bepaalt die beginnen met `start` waarbij `start` een nul of een één is. Hierbij is een rijtje goed als er geen twee nullen achter elkaar in staan.]